51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2013年03月
总第024期

2013年3月编程语言排行榜：有毒的Java

响应式Web设计是大势所趋还是时代的产物

观国内创业浅谈创业的三要素

目录

2013.03 Contents

51CTO.com
技术成就梦想

出版方 :北京无忧创想信息技术有限公司

责任编辑 :彭凡　林师授　张伟　　封面设计 :苍旭

联系方法 :zhangwei@51cto.com　010-68476606 (分机 8110)

出版日期 :2013 年 03 月 20 日

欢迎广大读者投稿

2013年3月编程语言排行榜:有毒Java

Tiobe 公布了新一期编程语言排行榜。Java 依旧是占据第一的位置，C 语言紧随其后。值得注意的 Objective-C 持续发力,已经占到了第三的位置。咋一看榜单,前 5 条中 C# 下滑最快,从第 3 名下降到第五名。而其他语言都与之前没有变化。

最近一段时间,关于 Java 安全性的新闻层出不穷。被伤害的不光是普通计算机用户,甚至还包括苹果公司、美国政府。此次安全风波波及面之广,恐怕是 Oracle 始料未及的。

1. 黑客利用网页漏洞进行攻击

据国外安全研究机构称,当前的 Java 版本中包含了一个严重的安全漏洞,可能导致电脑在访问带有恶意代码的特定网页时被感染。Window 系统中的所有主流浏览器都可能被利用并感染,其中也包括 Chrome。该漏洞也存在于苹果最新的 Mountain Lion 操作系统。

为此,苹果不得不单独发布 Java 相关更新,以保证 Mac 用户的安全性。素以跨平台性强著称的 Java,也让所有运用他的人无处可藏。

2.Java 安全机制

Java 早期的安全框架强调的是通过验证代码的来源和作者,保护用户避免受到下载下来的代码的攻击。

Java 安全模型的前三个部分——类加载体系结构、class 文件检验器、Java 虚拟机(及语言)的安全特性一起达到一个共同的目的 :保持 Java 虚拟 机的实例和它正在运行的应用程序的内部完整性,使得它们不被下载的恶意代码或有漏洞的代码侵犯。相反,这个安全模型的第四个组成部

Position Mar 2013	Position Mar 2012	Delta in Position	Programming Language	Ratings Mar 2013	Delta Mar 2012	Status
1	1	=	Java	18.156%	+1.05%	A
2	2	=	C	17.141%	+0.05%	A
3	5	↑↑	Objective-C	10.230%	+2.49%	A
4	4	=	C++	9.115%	+1.07%	A
5	3	↓↓	C#	6.597%	-1.65%	A
6	6	=	PHP	4.809%	-0.75%	A
7	7	=	(Visual) Basic	4.607%	+0.24%	A
8	9	↑	Python	4.388%	+1.10%	A
9	13	↑↑↑↑	Ruby	2.150%	+0.74%	A
10	10	=	Perl	1.959%	-0.74%	A
11	8	↓↓↓	JavaScript	1.370%	-2.02%	A
12	48	↑↑↑↑↑↑↑↑↑↑	Bash	1.009%	+0.78%	A-
13	15	↑↑	Lisp	0.942%	+0.02%	A
14	12	↓↓	PL/SQL	0.921%	-0.50%	A-
15	11	↓↓↓↓	Delphi/Object Pascal	0.889%	-0.84%	A
16	16	=	Visual Basic .NET	0.888%	+0.10%	A
17	14	↓↓↓	Transact-SQL	0.836%	-0.09%	A-
18	17	↓	Pascal	0.697%	-0.07%	A-
19	21	↑↑	Lua	0.697%	+0.17%	B
20	26	↑↑↑↑↑↑	Assembly	0.633%	+0.21%	B

分是安全管理器,它主要用于保护虚拟机的外部资源不被虚拟机内运行的恶意或有漏洞的代码侵犯。这个安全管理器是一个单独的对象,在运行的 Java 虚拟机中,它在对于外部资源的访问控制起中枢作用。

Java 签名 / 证书机制,可以保障使用者,安全地调用外部提供的 jar,防止你信任的 jar 被篡改。首先,Java 的签名,必须是基于 jar 包的。

也就是说,你必须将你要提供的 class,打包到 jar 里。然后,通过 Java 提供的签名工具(jarsigner)对 jar 包进行签名,发布。

签名原理 :用非对称算法,生成一对公钥 / 私钥。

2013 年 3 月编程语言排行榜 :有毒的 Java

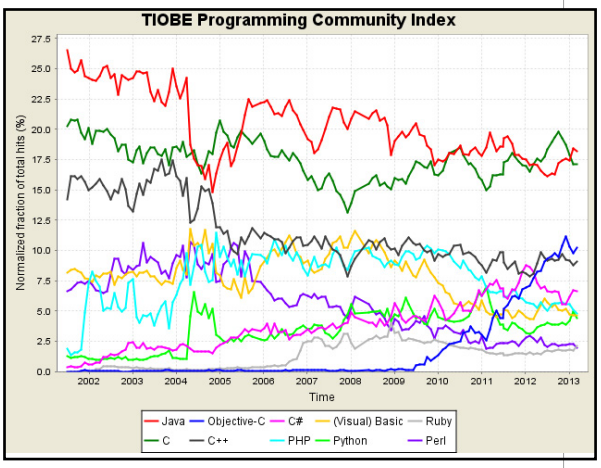
3.Oracle 应对 Java 漏洞危机

甲骨文软件质量保证经理Eric Maurice透露，五个漏洞中有四个存在于桌面系统的Java Web Start 应用和浏览器Java 小程序中。其中三个被通用漏洞评分系统评为 10 级，这是很严重的事情 :如果Java 运行在 Windows XP 这种默认以管理员身份运行的系统，那么黑客能够利用这些漏洞完全损害系统的保密性，完整性和可用性 ; 在Linux 或者 Salaris 这类以非管理员权限运行的系统中情况会好一些。安全研究人员对剩下的一个漏洞也做出过说明，该漏洞影响服务器部署的Java 安全套接字扩展 (JSSE)，基于 Lucky Thirteen 攻击 SSL/TLS 实现。

新的Java 6 _update41 可以从甲骨文网站下载，而不是Java.com，目前必须手动下载。但Java 6 安装程序的更新功能会提示用户下载并安装Java 7 _update15。这一切都在甲骨文的计划中。甲骨文之前在网站上宣布，将启动自动更新帮助Windows32 位系统用户完成升级。甲骨文将加快其对Java 的修补速度。Maurice 说，“甲骨文会继续加快Java 更新发布速度，特别是帮助解决桌面系统浏览器的Java 运行环境安全，以重树安全信誉。”

其实这次Java 漏洞危机并不是第一次，之前2010 年也有报道宣称Java 漏洞会影响 Windows运行安全。各位用户还是尽量及时更新自己的Java，作为开发语言中举足轻重的语言，Java 的安全性还是值得信赖的。 本期编程语言排行榜的其他排名数据和趋势走向。

前 10 名编程语言走势图



下面是第 50 到 100 的编程语言排名

(Visual) FoxPro, 4th Dimension/4D, ABC, Agilent VEE, Alice, Apex, AutoIt, AutoLISP, bc, C shell, CL (OS/400), Clipper, Clojure, Dart, Dylan, ECMAScript, Eiffel, Emacs Lisp, F#, Go, Groovy, Icon, IDL, Inform, Informix-4GL, J, JScript.NET, Ladder Logic, LPC, MEL, MUMPS, NATURAL, Oberon, OCaml, Occam, OpenCL, Oz, Pike, Q, REXX, S, sed, Simula, Smarty, SPARK, VBScript, VHDL, WebDNA, xBase, XSLT

下面给出了编程语言类别的一年变化趋势

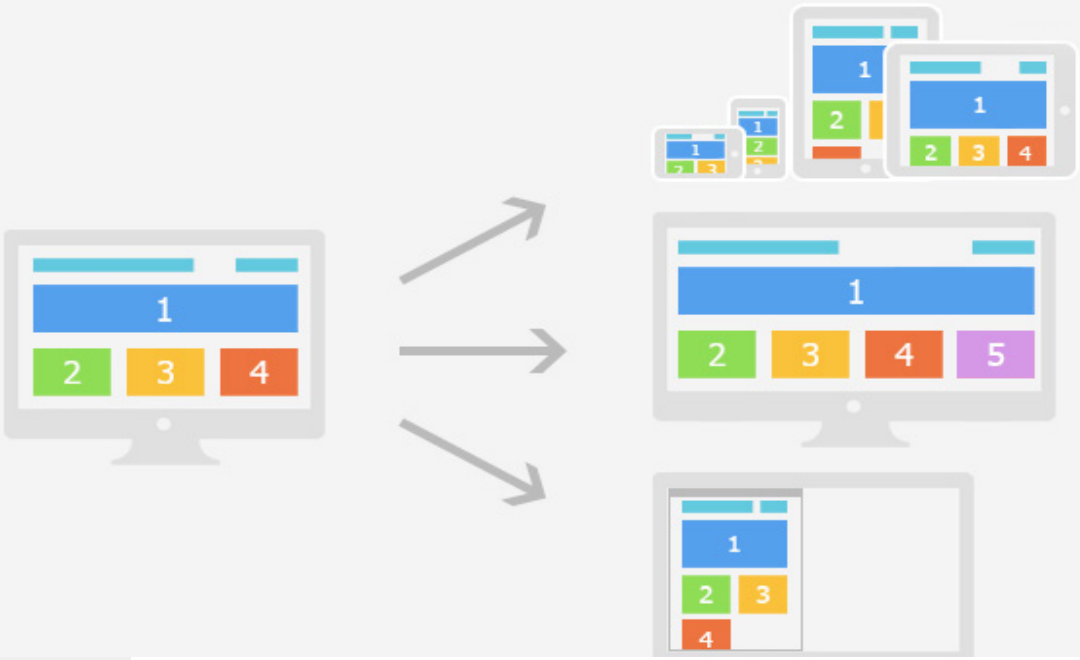
Category	Ratings Mar 2013	Delta Mar 2012
Object-Oriented Languages	60.4%	+3.0%
Procedural Languages	34.7%	-1.3%
Functional Languages	3.1%	-1.1%
Logical Languages	1.8%	-0.6%

Category	Ratings Mar 2013	Delta Mar 2012
Statically Typed Languages	71.0%	-0.1%
Dynamically Typed Languages	29.0%	+0.1%

更多内容，请查看原文，链接 :

http://developer.51cto.com/art/201303/384174_1.htm

■



响应式 Web 设计 (Responsive Web design) 的理念是页面的设计与开发应当根据设备环境 (屏幕尺寸、屏幕定向、系统平台等) 以及用户行为 (改变窗口大小等) 进行相应的响应和调整。具体的实践方式由多方面组成,包括弹性网格和布局、图片、CSS media query 的使用等。无论用户正在使用 pc、平板电脑,或者手机,无论是全屏显示还是非全屏的情况,无论屏幕是横向还是竖向,页面都应该能够自动切换分辨率、图片尺寸及相关脚本功能等,以适应不同设备。

响应式 Web 设计的优势 :

开发、维护、运营成本优势 :页面只有一个,只是针对不同的分辨率、不同的设备环境进行了一些不同的设计,所以在开发、维护和运营上,相对多个版本,能节约成本。

兼容性优势 :移动设备新的尺寸层出不穷,定制的版本通常只适用于某些规格的设备,如果新的设备分辨率变化较大,则往往不能兼容,而开发新的版本需要时间,这段时间内的访问就是个问题,但是响应式 Web 设计可以提前预防这个问题。

操作灵活 :响应式设计是针对页面的,可以只对必要的页面进行改动,其他页面不受影响。

51CTO 开发频道语

响应式Web设计是大势所趋还是时代的产物

在你身上是否发生过如此一件事，就是在你休息的时间用浏览器最大化的看一些文章、玩一些网页游戏正入迷的时候，时间一点点的过去了，马上到了下午工作的时间了，却发现还差一点就看完，于是乎，你把浏览器缩小了一些尺寸，瞬间，网页进行的重组，最后你发现你想看的内容就这样消失了，或是以一种难以阅读的方式呈现在你面前。这时你会希望，这要是响应式 Web 设计…该多好。

现今，无论是移动设备还是平板电脑亦或是 PC 屏幕，都有着各异的屏幕大小，若是针对每个屏幕大小单独设计一个解决方案会显得如此的繁琐和笨拙。响应式 Web 设计的概念就是可以让网页根据用户的使用环境进行自动调整，并有效的解决掉了用户体验度这一问题。

如此一说，响应式 Web 究竟是现今的大势所趋，还是因为各种设备的产生而诞生出的一种时代产物呢？下面，我们先了解一下响应式设计。

传统网页如何变成响应式

响应式 Web 设计已经成为目前主流的设计模式，那么对于那些以前固定好宽度和像素的页面来说，很多人都在想应该如何让它实现响应式？是否要进行网页的重构？那么传统网页如何变成响应式设计呢？ 51CTO 的记者从采访中了解到，在传统的 web 设计中，就提倡在 CSS 文件里将页面的布局、颜色、字体设计等分开，如果在当初进行设计时遵守了这一原则，那么在响应式 web 设计中，只需要更改页面布局这一部分的 CSS 内容即可。对于固定好宽度和像素的页面，也只需要按比例换算成百分比模式便可。

响应式 Web 设计的基本原理就是宽度使用百分比制以及利用 CSS3 Media Queries 语句事先设定各种分辨率下的页面布局。相当于对同一个网页，设计师要进行好几份设计（例如宽屏下的三栏设计、普通屏幕下的两栏设计以及移动设备上的单栏设计等）。

但现在似乎还没有一个比较统一的 CSS 框架或设计模式来减少 CSS 设计师的工作量。

对于是否需要进行页面重构，这就需要看网页在“裸奔”情况下是否可以有序的展示，如果在 CSS 无效的情况下网页无法有序地展示内容，那么是必须重构的，其实这跟是否进行 web 响应式设计无关，是对页面结构的最基本要求。

一般而言，出于展示的原因，移动设备可能不太适合诸如内嵌网页地图等元素，这就需要在 CSS 文件里面做相应的屏蔽设置。另外，若从用户的手机流量方面考虑的话，大页面最好还是进行重构，以减小用户向服务器请求的数据量。因为过大的数据请求一方面会增加用户在手机流量上的开销，更重要的一方面会降低页面的加载速度，导致移动设备的用户体验变差。

浅谈响应式 Web 设计的用户体验

一个网站若是没有一个良好的用户体验度，那么就算里面的内容在诱人，也无心阅读下去。

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201303/383855.htm

编者按

对一般创业者来说，也许他们无法做到在细节上让人觉得尽善尽美，也无法使功能上做到面面俱到，他们只要认真研究透用户心理，只要能提供一款满足用户兴趣的产品，这就已经足够。

观国内创业：浅谈创业三要素

【51CTO 专稿】互联网没有奇迹，如果创业者在追求上太过于完美，更是打算拥有一款独一无二的产品计划，那都是幻想。如果没有超越的激情与信仰，没有出色的产品运营，没有完整的管理制度，那么创业者将为自己的产品服务担负极高的成本风险、资金、各企业的模仿打压等等这方面的因素，最后都有可能使创业者的心血付诸东流。

对一般创业者来说，也许他们无法做到在细节上让人觉得尽善尽美，也无法使功能上做到面面俱到，他们只要认真研究透用户心理，只要能提供一款满足用户兴趣的产品，这就已经足够。下面来谈谈创业必要的三要素。

激情与信仰

众所周知，成功是无数的失败铸就。然而激情就是驱动创新梦想的助燃剂，每一次的失败都转换成为动力，为了达到自己的目标，了解专业领域知识，认识到自己的重要价值。如果没有激情，没有信仰，不敢去实践，不敢去面对失败，就会坚持不到最后。

缺少信仰的激情也是一样，就像射箭找不到靶子。所以，激情并非成功的唯一要素。一个成功的创业团队，激情和信仰是相互匹配的。但要想在缺乏激情的情况下打造成功的公司会更为困难。如果你把激情和信仰并在一起，这就会是创业公司成功的助推剂。

所以，有激情才有信仰，有信仰才有成功的动力。

产品与运营

在中国这个互联网行业里面，也许会有人认为只要有一款好的产品出现，立马就会出现无数的模仿者，在众多的模仿者中的一些大家伙，如腾讯、百度往往都会随时插上两腿，这会是众多创业者们的噩梦。其实不然，像腾讯这样的大家伙，它们是不会愿意把自己的核心团队去模仿一个新的产品，更多的是考虑让二线的开发团队去开发。作为一个创业团队何惧之有？

这个市场是瞬息万变的，而用户的需求也是在一个动态调整的过程，好的产品已经不是壁垒，在现在的环境下我们的重心不是这个产品的模仿者出现，更多的是我们应该把产品和自己的商业模式建设好，有效的管理自己的公司团队，跟进用

户对产品的需求变化。

所以任何的产品不能太理想化,如果缺少对产品的远景规划,没有把握住市场机会,没有建立起商业模式,再好的产品你也会甩出去。虽然心存侥幸手握 50W 的用户招来 VC 的目光,产品的用户没有意愿花钱,本身也缺乏广告的价值,人家凭什么投资?所以天天在埋怨别人模仿,还不如运营好自身的产品。

管理与制度

创业的基础在于人才,但更需要一套好的管理方式。如果陈旧的管理,没有制度的变革,就像 108 好汉各为其政,各行其是,整个公司更不易掌控。知名企业 Intel, IBM,微软等等无不有着全套完整的管理体系,相应的制度。

在中国创业团队的管理实践中,很多地方都是相互矛盾的,但却又是互相兼容的现象,这些东西实在是让人费解与彷徨。例如,

1)管理者从来不去实践,却认为自己完全掌握了管理的真谛所在;

2)不依照事实,自创管理方式,不依照传统,却强加其根本改变,本质的变更使管理混乱;

3)只看结果而不注重过程,守株待兔,一旦公司跨了,大势已去;

4)在领导面前努力做爷爷,树立威严,殊不知爷爷也是从孙子做起的;

5)强调公司战略,放眼未来管理模式,而不立足于现在需要解决的问题。

所以,创业需要务实,把自己全新投入到过程当中,那么这个过程自然你就会引领你向前发展。管理需要如此,企业更需要如此。■

响应式网页设计基础：灵活性

在过去的一年里，如果你不是住在深山里，就一定知道响应式网页设计，它已经成为当今的主流。响应式设计是 Ethan Marcotte 提出的，概念很简单：使网站的页面布局能够根据不同设备和分辨率进行自动调整。

当我第一次了解到它时，我就立即被迷住了 -- 特别是 media queries，我马上就用到我自己的个人兼职网站上。我甚至写了一篇文章介绍《如何使用 CSS3 Media Queries 响应不同设备》强烈建议在读这篇文章之前读一下）。 在第一次尝试使用 media queries 后，我很快意识到我忽略了一个响应式设计的重点：灵活性。

挑战固定宽度

我的个人兼职网站使用了固定宽度的设计，所有的 width, margin 和 padding 都使用了固定的像素值。 我一般都会这样写网站，因为对我来说它更简单，更快速。

但当我试图在我的固定宽度的网站上应用 media queries 的时候，那些简单和快速的优势全部都消失了。为什么？因为对于固定宽度的设计，我需要非常细致的定义 media queries 并在 CSS 文件中调整每个单独的像素值，基本上，我需要为每一种可能的分辨率都设计一个全新的布局，繁琐！ 慢!! 还不好玩!!!

我有幸听了 Marcotte 先生在《In Control 2011》的演讲，他讨论了响应式设计的理论和最佳实践，诸如 fluid grid（流体网格）实现方式。

流动且灵活的公式

流动式布局是灵活的。由于 width，margin 和 padding（甚至字体和图像）使用了百分比和 em（相对长度单位），因此页面布局会随着浏览器的窗口变化而变化。随着分辨率的改变，布局会成比例地进行调整，实现过程中不需要用到任何 media query。

这对于实现响应式网页设计来说简直太美妙了。如果我有一个基于比例值的布局，流动式的网格将替我完成大部分繁重的工作。我的 media query 将不再需要包含那些覆盖其他分辨率的所有 width, margin 和 padding 的样式定义。

但是也有一点让我感到头疼，计算流动式网格的比例宽度需要用到一些数学知识，我数学不太好…

幸运的是，Ethan 提供了一个很简单（即便对我来说）的公式来计算比例宽度：

目标宽度 ÷ 上下文宽度 = 结果（比例宽度）

这个公式用子元素的像素宽度（目标宽度）除以它父元素的像素宽度（上下文宽度），得到的结果就是这个子元素的比例宽度。

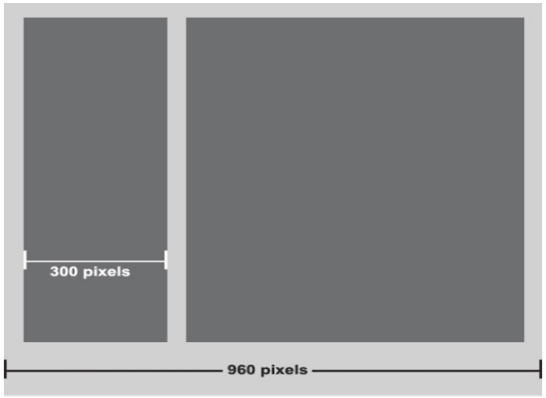


图 1 实例：目标宽度（300px）

响应式网页设计基础 :灵活性 II

和上下文宽度(960) 像素 。

在图 1 中 , 例如 , 深灰色区域宽度为 300px, 包含在宽度为 960px 的浅灰色区域中。

这里 ,960 像素区域是上下文元素 ,300 像素区域是目标元素 , 所以我们的数学公式是 : 300 ÷ 960 = 0.3125

0.3125 这个结果需要变成浏览器可识别的数值 , 因此需要转化成一个比例值 , 将小数点右移两位, 变成 31.25% 即可。然后在 CSS 中 , 将元素的宽度设定为这个比例值 :

```
aside{
background-color:#ccc;
float:left;
width:31.25%;
}
```

测试一下

公式虽然看起来很简单, 但我知道必须在实际的网站中检验一下才行。幸运的是, 我最近加入了 EE 播客, 正在重新设计那个网站。当我的搭档给我她的 PS 设计文件时, 我就决定将它打造成灵活布局的网站。

比例宽度

我首先记录下所有元素的像素宽度。(在排版设计事, 我们没有严格遵循网格布局, 这也是我建议的做法) 正如你在图 2 中看到的。整体宽度为 940 像素, Logo, 主持人介绍和分享链接都有它们各自的像素宽度。

按照 Ethan 的公式, 整体宽度 940px 就是我的上下文宽度, 根据它就可以确定所有元素的比例宽度 .

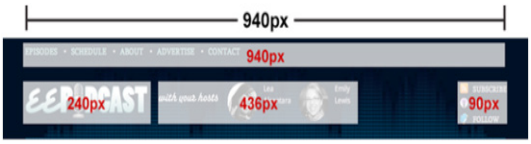


图 2 主页和顶部导航元素的像素宽度

Logo: 240 ÷ 940 = .255319148

主持人介绍 : 436 ÷ 940 = .463829787

分享链接 : 90 ÷ 940 = .09574468

随后我将这些浮点值转换成百分比值, 运用到我的 CSS 中 :

```
#logo a{
width:25.5319148%;/*240px/940px*/
}
#hosts{
width:46.3829787%;/*436px/940px*/
}
#push{
width:9.574468%;/*90px/940px*/
}
```

我没有对这些百分比值进行四舍五入, 而是直接运用在 CSS 中。我也从来没有遇到过任何浏览器对这样精度的百分比值识别错误的情况 (包括 IE)。

同时, 我为每一个百分比值都注释了计算它所用到的目标宽度和上下文宽度, 这对于今后的开发来说是非常重要的参考。

■本文未完, 完整部分请浏览

http://developer.51cto.com/art/201302/381358.htm

响应式网页设计与应用

响应式 Web 设计 (Responsive Web design),理念是设计和开发应根据屏幕的大小、平台的用户的行为和环境基础上自动调整 ;他应该有一个灵活的网格和布局,图像和 CSS 能够智能的组合使用。

每个设计师都想自己的网站可以在任何设备上体验都很牛,神马 iPhone、ipad、黑莓、kindle……无所不能啊 ~ 苦逼的是这些东东的系统不一样,分辨率也不一样啊 ;电脑的屏幕越来越大,但是还有一半的用户还是用的 17 寸 CRT ;手机、pad 层出不穷,没有个统一标准,我们又不想失去任何一个用户,这可苦了我这些设计师了,需要付出更多的心血来获得更好的体验,谁让我们是射击湿呢。

电子产品更新换代节奏都快成马了,跟都跟不上。对于网站来说根本不可能为每种设备都开发个版本出来,结果就是赢得鱼,却失去了熊掌,不过我们还是有办法的。

响应式 Web 设计 (Responsive Web design),理念是设计和开发应根据屏幕的大小、平台的用户的行为和环境基础上自动调整 ;他应该有一个灵活的网格和布局,图像和 CSS 能够智能的组合使用。比如说用户从电脑切换到 ipad,网站能够自动切换以适应分辨率,图像大小和脚本。换句话说,网站应该具备根据用户的使用环境来自动调整,这可以减少不必要的重复设计。

响应式 Web 设计的概念

Ethan Marcotte 曾经在 A List Apart 发表过一篇文章“Responsive Web Design”,文中援引了响应式建筑设计的概念 :

所谓响应式建筑设计就是设计师尝试建造一种使用一些传感器检测周围环境,比如说温度、湿度、光线等等自动进行调整的房子。现在我们把这个思路延伸到 WEB 设计领域。我们可以想,为啥我们要为每个用户群各自打造一套设计方案呢?我们太笨了,有没有更智能的做法?和响应式建筑设计一样,web 设计也应该做到智能调整。

显然 web 设计不能使用传感器,这就要更多的抽象思维。好在现在一些概念已经得到实践了,比如液态布局、帮助页面重新格式化的 media queries 和脚本等。但是响应式 Web 设计不仅仅是关于屏幕分辨率自适应以及自动缩放的图片等等,它更像是一种对于设计的全新思维模式。

调整分辨率

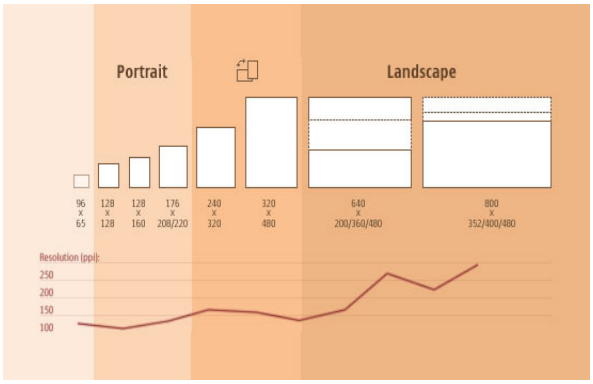
不同的设备都有各自的屏幕分辨率、清晰度以及屏幕定向方式,不断被研发的各种新设备也将出现新的屏幕尺寸规格。有些设备基于横屏 (portrait),有些是竖屏 (landscape),甚至还有正方形 ;对于日益流行的 iPhone、iPad 及其他一些智能手机、平板电脑,用户还可以通过转动设备来任意切换屏幕的定向方式。怎样才能做到让一种设计方案满足所有情况?

要想做到同时兼容横、竖屏 (用户还有可能在页面加载的过程中切换方向),我们就必须考虑 N 种屏幕尺寸规格。

响应式网页设计与应用 II

我们可以将这些规格划分为几个大类，然后为每一类做一种方案，确保该方案至少在本组中尽量具有弹性。但即使这样，结果也将是非常不爽的，谁知道某类设备在 5 年之后的占有率是多少？而且很多用户甚至不去将浏览器的窗口最大化；天哪，哥受不了了……

Morten Hjerde 和他的哥们对 2005 至 2008 年市场中的 400 余种移动设备进行了统计（查看报告），下图为大致统计结果：



在 08 年之后，更多更有代表性的新设备问世并普及了。显然，我们不可以沿着“多方案”的思路继续走下去；那么我们应该怎样做呢？

部分解决方案：一切弹性化

几年前，弹性布局 (flexible layout) 几乎是一种奢侈品，所谓弹性，也只是体现在竖排布局以及字号等方面；图片尺寸无法变化，这就导致图片破坏了页面结构，而且即使是有弹性的元素，在很多情况下他会乱弹，仍然是一塌糊涂。所以，所谓的弹性布局其实并非那样弹性，它有时甚至不能适应台式机与笔记本的屏幕分辨率差异，更不用说手机等移动设备了。

现在，我们可以通过响应式的设计和开发思路让页面更加”弹性”了。图片的尺寸可以自动调整，页面布局再不会被破坏。虽然永远没有最完美的方案，但它给了我们更多选择。无论用户切换设备的屏幕定向方式，还是从台式机屏幕转到 iPad 上浏览，页面都会真正的富有弹性。

在前文提到的 Ethan Marcotte 的文章中，他通过一个实例展示了响应式 Web 设计在页面弹性方面的特性：

该实例的实现方式完美的结合了液态网格和液态图片技术，并且聪明的在正确的地方使用了正确的 HTML 标记。”液态网格”是一种很常见的实现方式。

说到创建液态页面，最好看看 Zoe Mickley Gillenwater 的那本《Flexible Web Design: Creating Liquid and Elastic Layouts with CSS》“怎样创建弹性图片”。另外，Zoe 的这篇”Essential Resources for Creating Liquid and Elastic Layouts.“提供了不少关于创建弹性网格和布局的教程、资源、创意指导和实现方式。

从技术角度讲，虽然听上去这些都很简单，但它着实相当复杂。举个例子，仔细观察 Ethan Marcotte 提供的实例中的 logo 图片。

■本文未完，完整部分请浏览

http://developer.51cto.com/art/201303/382733.htm

编者按

众所周知，所谓响应式页面，就是能够用一套样式，使你的页面能够在不同分辨率的屏幕下都有很好的表现形式。响应式 Web 设计，这个概念是 Ethan Marctte 在 A List Apart 发表的一篇文章“Responsive Web Design”中援引响应式建筑而得名。

关于响应式页面

作为一个无线部门的人，不懂移动设备是不行的。而作为一个无线的重构，不会写响应式页面更是不行得。而我，一个无线的重构，在我最近做的一个移动端的项目之前，的确是不会写响应式页面的，所以，严格来说，在这个项目之前，我是一个不合格的无线重构人。

而这个项目，却让我快速地掌握了响应式页面重构的一些方法。下面就是通过这个项目来总结我在响应式页面重构学到的东西。

众所周知，所谓响应式页面，就是能够用一套样式，使你的页面能够在不同分辨率的屏幕下都有很好的表现形式。

响应式 Web 设计，这个概念是 Ethan Marctte 在 A List Apart 发表的一篇文章“Responsive Web Design”中援引响应式建筑而得名：

响应式建筑（responsive architecture），物理空间应该可以根据存在于其中的人的情况响应。

根据我所阅读过一些文章及资料，我总结出响应式页面的几个关键组成部分：

1、页面头部的 meta 说明，可以通过 viewport meta 标签去让你的 html 页面的的宽度能根据设备分辨率让浏览器的可视宽度来适应。

也可以在这里设置页面的缩放比例等等，这样在成比例的分辨率设备下，就可以更简单地实现响应式。

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

2、流体布局（fluid grid），所谓的流体布局，其实就是在你 pc 端实现的页面基础上，将一些元素的宽高由原来的固定多少像素（px）调整为百分比（%）或字体比例（em）（或布局方面的 margin、padding、left、top 等以 px 为单位的值），这也是当前实现响应式布局的两种主要实现方法。

第一种用百分比（%），就是以该元素的父容器的宽高为 100%，其他元素的宽高相对于其父容器的比例，只要将具体的像素值相对于他的父容器的一个百分比折算即可。当然这种方法的换算有点复杂，因为很多相对的宽高折算的百分比系数是带小数的，所以这时候可能要你有足够的耐心才能实现。

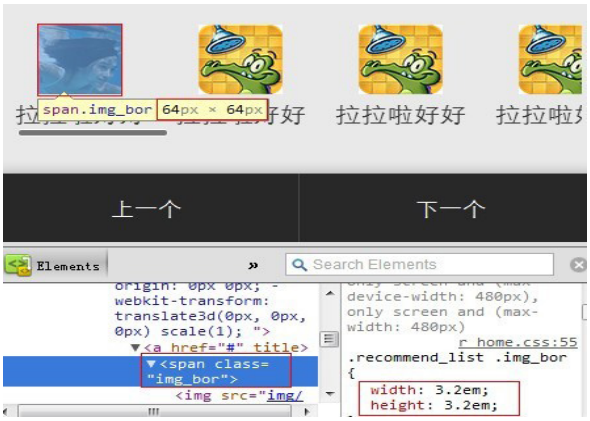
在 Ethan Marctte 的 Responsive Web Design 这篇文章中给出的一个 demo 中，我们可以看到他的实际代码里：

```
@media screen and (max-width: 400px) {
.figure,
li#f-mycroft {
margin-right: 3.317535545023696682%; /* 21px / 633px
*/
```

关于响应式页面 II

```
width: 48.341232227488151658%; /* 306px / 633px */}
```

第二种方法是用字号比例(em)去实现,其实方法是跟上面一样的,只不过我们将 % 换成了 em,这种方法就是某元素具体的宽高(px)在当前基准字号(font-size)下折算出多少个 em。eg : 一个在 480 分辨率下宽高为 64px*64px 的元素,其父容器的字号(font-size) 为 20px,那么它折算成 em 为单位就是 3.2em*3.2em。当其父容器字号基准根据不同的分辨率变化的时候,该元素的宽高也能根据这个字号基准成比例的缩放,就能实现响应式变化。



从上面的两张实例图我们可以看到,同一个元素,宽高为 3.2em*3.2em,在 360px 分辨率下,因为基准字号为 15px,故解析出来的实际尺寸为 48px*48px,而在 480px 分辨率下,基准字号为 20px,故实际的尺寸为 64px*64px。

3、流体图片(liquid image),在我所了解的很多资料中,对图片处理这块,如果要使图片能根据分辨率来适应,而且还不失真,好像挺困难的。但其实我们不用考虑的那么复杂,我们要做的只是让图片能根据不同分辨率自适应罢了,我们不管图片会不会因为被拉伸而失真,因为真的遇到这样的情况,我们可以考虑在不同分辨率下使用不同的图片,这样就简单多了。所以让图片尺寸自适应,我们只要不给图片设定具体的宽高尺寸,只要在样式中给该图片一个 width :100%,这样图片就能根据它父容器的尺寸自动调整了。

4、媒体查询(media query),这个也是响应式页面的一个关键技术,根据不同的分辨率去调整一些不同的样式。

```
@media screen and (max-device-width: 480px) {
.column {
float:none;
}
}
```

通过上面的这样媒体查询结构,我们可以设定在不同分辨率下选用不同的样式来调整响应式页面。像前面第二点流体布局上,我们使用百分比或字号比例去实现流体布局的时候,第一种方法是可以不用媒体查询直接实现流体布局的,就是元素的宽高能自适应不同分辨率屏幕。■未完

http://developer.51cto.com/art/201302/380491.htm

源代码管理十诫

若是还有可以毫无偏见地涉及各个编程语言，比源代码管理软件更必要的工具，我倒是很想见识一下。源代码管理软件是我们工作的必备工具，是许多开发团队的血液。那为什么我们都会对它有所误解呢？为什么都很难理解版本控制系统的核心价值和基本原理呢？

我总结出 10 条惯例——如果你愿意也可以用“戒律”——意味着必须服从它而且从一开始很难去理解。它们与所有类型编程语言的版本控制软件都有关联。在这里我选取了 Subversion 和 .NET 的几个例子，不过它们也广泛地适用于其他的一些技术。

第一诫 . 如果你现在还在使用 VSS – 请立刻停手

它已经死了。当然不完全对，它也存活了许多年，被全新的更实用的源代码管理工具超越之后还在苟延残喘地活着。准确地说当微软几个月后不再为其提供支持时（还是会坚持一段时间的），它才是真的死了。

平心而论，VSS 还是一个不错的工具。在 1995 年，它的光芒被像 Subversion 这样类似于 Git 和 Mercurial 的分布式软件给遮盖住了。微软表示要取代它已经好多年了。

原因是因为不支持如今的标准所导致的一系列缺陷使它一直不被看好。众所周知它是微软的悲剧系统，但不知何故它能坚持这么久，尽管它有那么多小故障，缺陷，并且不包含必需的功能（相对于今天的标准）。

第二诫 . 如果代码没放在源代码管理软件里，等于它不存在

每天重复读这句话——“使用源代码管理软件是唯一的有效措施”。除非你在工作时使用项目的源代码管理库来控制代码版本——否则代码等于没有存在过。

显然你曾发觉在你的本地机器上运行良好的代码在其他人那里运行的效果并不理想。是不是？他们不能获取你的最新版本，他们没法去归并代码文件，你没有正确地部署它（参考 you're deploying it wrong）而且如果你的 SSD 硬盘坏了的话你将永远地失去你的劳动成果。

只要你保持这个心态——代码只有提交后才是真的安全，才是其他良好编程习惯的保障。你可以把你的任务划分成许多很小的单元以便你逐一提交。你需要频繁地这么做。你就不必担心你的硬件会不会出棘手问题。

不过更重要的意义是（至少对于你的团队领导来说），通过源代码管理软件可以看到你做了什么。使用图表并列出项目清单是个好方法，不过怎么知道他们实际上在做些什么？而使用源代码管理软件进行工作就能看得一清二楚了。

第三诫 . 要早提交，常提交，并且不要觉得麻烦

关于前面那点，避免“幻影代码”（就是只能在你的机器上看到的代码）的唯一方法是经常提交你的任务并且不要觉得麻烦。它可以解决你的问题，不过这样做也会对你的工作 ...

源代码管理十诫 II

产生其他的影响：

1. 每个提交的修订都会为你提供一个还原点。如果你完全把代码搞砸了(没骗你，我们都这么做过)，你是希望恢复到一个小时前的工作还是一周前的工作？

2. 归并文件时会出现的危险会随着时间不断增加。归并文件一直很麻烦。如果你不是每天都保持提交代码，某一天你会突然发现你和其他人的更改内容会有 50 多个冲突。你不会为此感到高兴的。

3. 它促使你把任务分离成分散的单元。通常人们都是快完成的时候才提交的，因为他们想把代码做成一个完整的逻辑单元模块。不过庞大的任务不可避免地要分离出较小的分散功能，而频繁地提交它们会使你更了解它们，你可以一个个地构建并提交。

如果你做到这些，你的提交历史不可避免地开始类似于一种半规律的样式，里面每个工作日都是在提交任务。当然不总是这样，也有停下来重构或测试，或者其他合理的活动也会中断标准的开发周期。

然而，当我在看一个独立的——尤其是完整的项目时，每当发现我们在一个标准的开发周期里，有一天或几天什么都没有做，我便会非常担忧。我之所以担忧是因为这意味着什么地方出问题了。一般不是有人正在想方设法要把问题搞定的话，就是因为卡在某个问题上而导致项目完全没有进度。无论到底是什么情况，源代码管理软件都会告诉你出现问题了。

第四诫 . 提交前要检查你更改了什么

往源代码管理软件里提交代码的步骤其实非常简单。(你恐怕会困惑上一条为什么说的那么麻烦。)一般只要发现文件内容有变更时都会不顾后果地把文件传上去。像这样——“我的项目根目录下有文件内容变更了，我要快点提交上去！”

如此会发生一件(或两件)事情：首先，程序员会没有意识地把目录下的垃圾代码文件也上传上去。一些人看到类似下面的窗口时，就会点击“选择全部”然后提交——

这样源仓库里就会被本不应该存在的未调试的文件和其他垃圾文件给弄乱。

或者是，程序员实际上并没有检查他们更改过什么就把文件上传了。当你在工作中处理配置文件或项目定义文件时很容易就不经意把那些不想提交的文件给上传了，而且那些文件很可能就被别的程序员用到了。你真的会记住你在配置文件里的所有更改吗？

解决方法很简单：你必须在提交前立刻检查你改过什么地方。做起来其实比听起来还要容易。使用许多系统已经提供的“忽略”特性可以大幅度地减轻“不经意上传文件”的危险。你可以忽略 Thumbs.db 文件因为你压根不想上传它。你在每次修订后可能还有其他文件不想上传——那么就忽略掉它们吧！

至于文件里的更改，你通常可以使用某个流行的文本比较工具来观察差异。为什么我又要上传一次 Web.config 文件呢？

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201303/383248.htm

网易财经前端开发总结

众所周知，http 请求是要开销的，减少请求数可以提高网页加载速度。常用的方法，合并 css，js 以及 Image maps 和 css sprites 等。

作为门户网站的前端，有许多说不出的苦楚：有些代码虽然自己也看不下去，但还是不得不硬着头皮把页面给拼上去，比如跟其他频道公用的部分：因为是公用，js、css 必须写在该部分，调用的 js 库（网易的很多频道头部都调用了两个大同小异的 js 库）等都身不由己。而且作为财经门户来说，页面分多屏是显然的、再者 N 多异步请求的数据（股票数据要及时更新）、画股票行情图等，对于前端的性能都是极大的考验。笔者用 YSlow 去测评了一下各大门户的财经频道，网易财经得了个 C，而老东家新浪为 F，腾讯、搜狐财经都 D（测试时间为 13 年 2 月 24 日），这点还是挺庆幸的，哈 ....

好了，废话不多说，下面笔者就 yahoo 的 14 条军规来总结一下网易财经的前端开发工作：

1、Make Fewer HTTP Requests

众所周知，http 请求是要开销的，减少请求数可以提高网页加载速度。常用的方法，合并 css，js 以及 Image maps 和 css sprites 等。笔者所在团队的做法是根据功能分开开发，然后通过内部系统对 js，css 进行分组合并，这样对于浏览器来说是一个请求，但是开发时仍然能还原成多个，方便管理和重复引用。值得一提的是，网易内部的前端代码发布系统是很值得学习、借鉴的，对于 fiddler 调试和性能优化工作很是方便，这里不方便透露更多，有兴趣的单独聊，哈。而 css sprites 是指只用将页面上的背景图合并成一张，然后通过 background-position 来取背景。这里笔者并不提倡看到像图片的都用图片来处理，就该项目的

这些部分，都是通过 css 实现的。作为程序猿，新技术我们要及时掌握并加以运用，尤其是所谓的大公司，更要有这种气魄。

不要因为某些极品的浏览器不兼容而放弃，庆幸的是，我们领导也很认同，哈哈～～

2、Use a Content Delivery Network (CDN 内容分发网络）

简单地讲，通过在现有的 Internet 中增加一层新的网络架构，将网站的内容发布到最接近用户的 cache 服务器内，通过 DNS 负载均衡（可选根据时间或访问速度来决定访问哪台服务器资源，笔者刚到不久，没有深入去研究公司这部分底层）的技术，判断用户来源访问 cache 服务器取得所需的内容。这样可以有效减少数据在网络上传输的时间，提高速度。相信这个很多公司都有做，这里就不多说了。

网易财经前端开发总结 II

3、Add an Expires Header

4、Gzip Components

当然，这几部分内容后端帮我们完成了，3 主要通过服务器端脚本设置 Cache-Control 和 Expires 完成 ；Gzip 的思想就是把文件先在服务器端进行压缩，然后再传输。这个压缩率很高，基本上可以压缩到原来的 1/4。笔者过去是 phper，所以对这块也略知一二，对于前端攻城狮，我们还是有必要了解这块的内容。

5、Put Stylesheets at the Top

我们知道，css，Cascading Style Sheets（层叠样式表）。层叠即意味后面的 css 可以覆盖前面的 css，级别高的 css 可以覆盖级别低的 css。。ie，firefox 等浏览器在 css 全部传输完全之前不会去渲染任何的东西。很多浏览器下，如 IE，把样式表放在页面的底部的问题在于它禁止了网页内容的顺序显示。浏览器阻止显示以免重画页面元素，那用户只能看到空白页了。Firefox 不会阻止显示，但这意味着当样式表下载后，有些页面元素可能需要重画，这导致闪烁问题。所以我们应该尽快让 css 加载完毕。实际上很多网站也是这么做的，当然有需要分屏显示的网站，为了让用户看到的首屏页面尽快显示，也可以分开放置，当然，这里要根据具体项目需求来讨论了。实际上笔者该项目也分了三个 css 文件，比如说是延时显示的广告，我们为了提高 css 加载速度，也独立出来了放在页面的底部。

6、Put Scripts at the Bottom

原因 :首先，防止 script 脚本的执行阻塞页面的下载。在页面 loading 的过程中，当浏览器读到 js 执行语句的时候一定会把它全部解释完毕后在会接下来读下面的内容。浏览器这么做的逻辑是因为 js 随时可能执行 location.href 或是其他可能完全中断此页面过程的函数，即如此，当然得等他执行完毕之后再加载咯。所以放在页面最后，可以有效减少页面可视元素的加载时间。其次，脚本引起的第二个问题是它阻塞并行下载数量。HTTP/1.1 规范建议浏览器每个主机的并行下载数不超过 2 个（IE 只能为 2 个，其他浏览器如 ff 等都是默认设置为 2 个，不过新出的 ie8 可以达 6 个）。因此如果您把图像文件分布到多台机器的话，您可以达到超过 2 个的并行下载。但是当脚本文件下载时，浏览器不会启动其他的并行下载。

7、Avoid CSS Expressions

尽量减少或者不使用 css 表达式，其实大部分可以用 js 实现。

8、Make JavaScript and CSS External

把 css 和 js 写在页面内容可以减少 2 次请求，但也增大了页面的大小。

我们的网站已经对静态文件做了缓存，那也就没有 2 次多余的 http 请求了。

9、Reduce DNS Lookups

一次 DNS 的解析过程会消耗 20-120 毫秒的时间，在 dns 查询结束之前，浏览器不会下载该域名下的任何东西。所以减少 dns 查询的时间可以加快页面的加载速度。yahoo 的建议一个页面所包含的域名数尽量控制在 2-4 个。这就需要对页面整体有一个很好的规划。目前我们这点做的不好，尤其是对于单纯靠广州收入的网站，很多广告投放系统拖累了我们。

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201302/381308.htm

用最快的速度设计一种新的编程语言

编者按

用最短的时间设计一种简单，但好玩的编程语言CShell（实现CShell解析器基本上用不着编译原理的知识，但以后的文章就会涉及到很多编译原理的内容了）。

最近打算写一些列有趣、而且有一定难度的文章。这个系列的名字就叫《疯狂极客》，这一系列的文章大多数与计算机有密切的关系。包括制作编译器、制作OS、Android控制电路板、机器人的制作（通过Android、IOS等设备控制）等等。

在正式开始《疯狂极客》系列文章之前，先来热热身。用最短的时间设计一种简单，但好玩的编程语言CShell（不过不用担心，实现CShell解析器基本上用不着编译原理的知识，但以后的文章就会涉及到很多编译原理的内容了）。从CShell的名称可以猜到，是一种C风格的语言，并且可以像Shell一样解释执行（动态语言）。当然，这种语言不可能像C语言或Shell一样强大，因为C语言的编译器实现起来尽管不复杂（因为是结构化编程语言，没有类、接口这些东西，实现起来要比Java编译器简单得多），但仍然不太可能在很短的时间内完成（一至两天）。不过本文实现的CShell尽管简单，但仍然可以实现一些算法。CShell语言支持输出值和变量、条件语句（if），for循环，自加、自减、+、-、*、/操作，函数（支持递归）。由于CShell是动态语言，所以不需要声明变量，不过支持全局和局部变量，当然，还支持数组（整数、字符串类型数组），所以使用CShell可以很容易实现像冒泡排序、阶乘等算法。

在讨论CShell的设计原理和实现过程之前，先看一些用CShell编写的程序。单从这些程序所完成的工作来看都太太太简单了，不过这回完全不同，这回是用我们自己发明的新语言来实现这些算法，例如，递归阶乘计算、冒泡排序，是不是很酷呢!!

如何设计和实现编程语言

设计一种编程语言的方法很多，当然，通常的做法是要学好编译原理，然后按部就班地从词法分析器做起，然后是词法分析器、语义分析、中间代码生成、中间代码优化，目标代码生成，如果语言需要使用runtime运行，还需要编写可以运行目标代码的虚拟机（解释目标代码的程序，例如jvm就是解析Java字节码文件的虚拟机）。看着就有点晕。而且估计现在很多科班出身的程序员编译原理学的一塌糊涂。就算编译原理学的很好，光凭编译原理的理论，如果要想编写一个比较复杂的编译器或解析器也是很难办到的（尤其是加入面向对象功能）。这是因为一个复杂的编译器有很多代码几乎不太可能完全通过手工编写，例如，语法分析如果使用LL（*）分析方式，计算大量的first和follow集合就非常恐怖，就算把代码编写完了。

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201303/384243.htm

缩短eclipse的启动时间的JVM优化

最近自从 eclipse 安装了很多插件以后，启动变得非常的慢，每次启动，要消耗近半分钟．这是不正常的．今天决定好好优化一下．

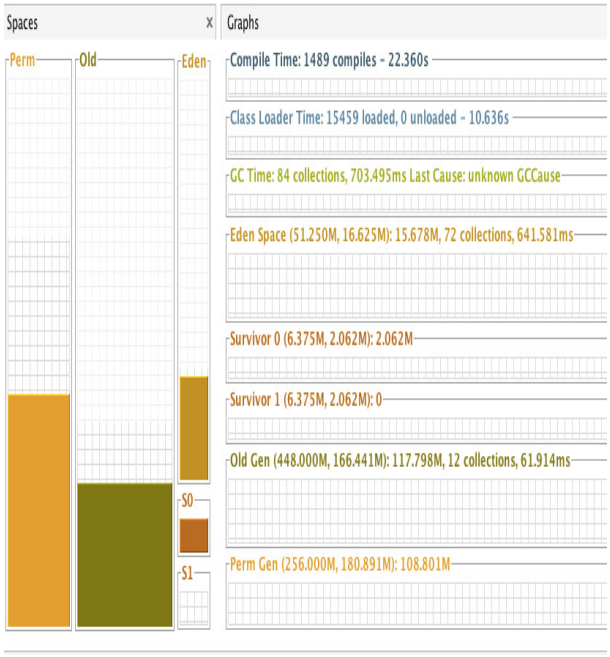
追加：首先要声明一下，这个案例在＜深入理解 JVM 虚拟机＞这本书中也提到过．这本书是我曾经学习 JVM 的第一本书．里面关于 Heap 的优化思想，来源于此．建议大家想学 JVM 原理的，可以找来此书看看．写这篇文章，是因为最近在给一个社交网站服务器做调优，突然觉得我机器上的 eclipse 跑的比较多，所以顺便优化下 eclipse. 至于基于 WebSphere 服务器的性能调优，这回涉及到更多的工具和方法，会在以后的文章中看到．

最近自从 eclipse 安装了很多插件以后，启动变得非常的慢，每次启动，要消耗近半分钟．这是不正常的．今天决定好好优化一下．

我所使用的 eclipse 是 Eclipse Java EE IDE for Web Developers 3.8 版本．跑在 MAC OSX 上，SSD+8G RAM, 这么高性能的机器竟然不能秒开 eclipse, 这太说不过去了．哦，还有我使用的 JVM 是 Oracle 的 HotSpot, 来自于 JDK1.6 64bit.

首先，在优化前，让我们看看 eclipse 启动时 ,JVM 的各项性能指标．因为我并不能准确的判定 eclipse 的启动完成时间，所以我只能说大约事件．

首先启动 JDK 自带的 JVM 性能监视工具，在 java\bin 的目录下，有一个 jvisualvm, 它是绑定在 JDK 中的 visualvm. 双击启动 visualvm. 然后启动 eclipse, 在 eclipse 启动完成以后，使用 visualvm 的查看 eclipse 的 Visual GC 情况，如图：

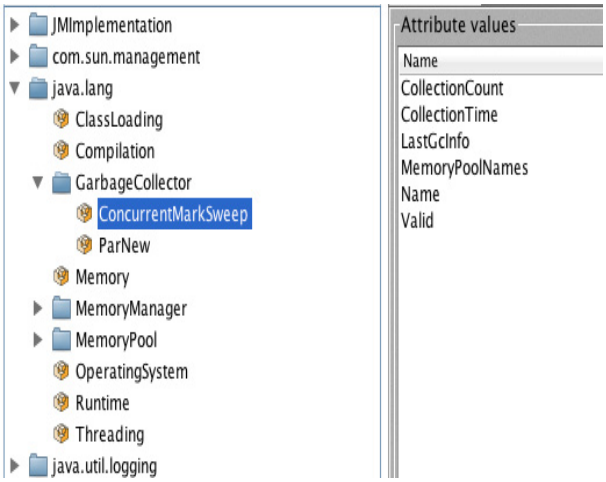


上图中说明在 eclipse 的启动过程中 ,JIT 对字节码进行了向机器码的编译，花去了 22 秒的时间 .Class 加载花去了 10 秒的时间 ,Minor GC 发生了 72 次，花去 0.64 秒 ,Full GC 发生了 12 次，仅仅花去了 61 毫秒．

我们再去 MBean 选项查看，发现新生代使用 ParNew 垃圾收集器，而老年代使用的是 CMS 垃圾收集器．

缩短 eclipse 的启动时间的 JVM 优化 II

总上情况看出，由于 MAC 的性能比较好，所以垃圾回收并没有消耗太多的时间，并且 CMS+ParNew 本身就是并行垃圾回收，不会造成用户程序太多的停顿．时间主要消耗在了 JIT 的即时编译和 Class 加载上了．



首先要优化的就是 class 加载．因为 eclipse 这个工具是一个成熟的工具，经过了这么多人的验证，所以我充分信任 eclipse 的代码，允许 eclipse 的代码在加载的时候，跳过字节码验证．关闭字节码验证的方法是在 vm 的 args 中加入参数 -Xverify:none. 对于 eclipse 来说，找到 eclipse.ini, 加入 -Xverify:none. 让我们再重启一下 eclipse, 看看 class 加载时间是否减小．再次启动，发现 class 加载事件缩小到 7 秒，比之前少了 3 秒．

然后优化的是 JIT 的时间．在使用 eclipse 编写程序时，主要是文本编辑，编译和运行，JIT 虽然可以带给我们高性能，但是 JIT 在编译机器码的时候，却要消耗很多的时间．eclipse 对项目的编译和运行本身就很慢，切运行时是启动一个新的 java 进程，跟 eclipse 本身无关，所以，我可以接受抛弃 JIT 编译器，而只是用 JVM 解释器执行字节码所带来的效率降低．这样可以去除 JIT 编译的时间．做法如下，在 eclipse.ini 中加入 vm 的参数 -Xint, 意思是只使用解释器．让我们来看看结果：

JVM 编译器时间变成了 0, 一下减掉 20 秒．但是，由于缺少了运行时的即时编译优化方案，代码的运行时间变长了，eclipse 的整体启动时间慢了更多，超过了 30 秒．由此可见，JIT 是多么有用的一项技术．所以禁止 JIT 的尝试失败了．我们把之前的参数 -Xint 去掉．

哦，对了，我还装了很多的插件，尤其是 android 开发插件．启动的时候对插件的激活也会花去很多时间．屏蔽插件激活的方法：Windows -> Preferences, 输入“startup”，点击“Startup and Shutdown”，把不需要的插件勾掉．此外，还需要关掉不必要的 validation, 方法为 :Windows -> Preferences -> Validation. 只选你需要的．

做完以上工作，我发现 eclipse 启动稍微快了一些．掐着秒表计算的花了大约 15 秒．

最后，再优化一下 GC 和堆栈吧．虽然说，GC 已经表现的很好了，都没有超过 1 秒，但是 GC 的频率如此高，说明 JVM 的内存的分配是不合理的．为此，我们需要重新对 JVM 内存进行划分．为了对 JVM 的内存进行合理分配，我们需要了解 eclipse 启动过程中 .... ■本文未完

http://developer.51cto.com/art/201303/382978.htm

个人开发者无需绝望

51CTO 记者的微博，前几日因为其他媒体一则“个人 APP 开发濒临绝境”的新闻，从而引发了一些讨论。文中的观点确实有几分道理，也是目前移动应用个人开发者的真实写照。比如创新遭遇技术瓶颈，有了营收又怕市场上其他公司都来山寨，辞职后专心开发所带来的生活压力等等

心态让个人开发者绝望

有位程序员就表示，开发者为了自己的钱包鼓鼓的，很多人都会昧着良心开始使用残次品了。所谓的“专业”是对产品负责，对自己的人格负责；所谓“商业”是对钱负责。目标不一样，出来的东西必然不一样。但个人开发者为了让自己能生存下去，向商业妥协也是权宜之计。只不过很容易陷进越商业，越残次的恶性循环。这样的个人开发者，能真正走出绝望的不多。

大公司真的是恶魔？

当我们开发出来的新应用，用户已经开始增长，甚至有了不错的营收之后，你就该担心企鹅这样的像素级创新公司来借鉴你的创意了。网名食匣子的微博网友就对 51CTO 表示“对于小开发者，背后没大树，简单的点子，很容易被大公司抄袭，生态环境也比较恶劣。”但同时他也指出“大公司有很多限制，研发一款产品，必须短期内就要见到效益，但很多产品，尤其是内容型产品是需要积累的。为了快速积累就抓取没营养的内容，反而降低了品牌的口碑。”这其实就给个人开发者提供了空间，选择大公司的缝隙求生存。

个人开发者没必要辞职

有的开发者确实怀着这样的思想，做事情是要隔绝外来的干扰的。一个产品的灵魂只属于一个人。比如《诛仙》，书不写完，是不跟别人签约的。这样精益求精的思想，确实是很多人追求的目标，但我们是不是要辞职后，专门来从事自己的开发创业呢？“可可私房菜”的开发者大头圆可可（新浪微博地址 :http://weibo.com/qiankeke ）有其自己的观点。

“我从头到尾都没有想过辞职从事专职开发，一直都是抱着业余时间做着玩的心态所以谈不上盈利和生存因为有自己的工作。”

——大头圆可可

无心插柳柳成荫的做法，确实能让开发者减小很多不必要的压力。生活上可以透过工资获得保障，公司的技术培训可以让自己的项目更加有技术竞争力，同事之间的技术交流也能给自己带来更多的灵感。在开发工具日趋智能化的今天，创意反而成为制约个人开发者的主要因素。在一种衣食无忧的状态下想想自己的产品，或许能获得更多的灵感。■

CSS 小技巧

attr() 功能早在 CSS 2.1 标准中就已经出现，但现在才开始普遍流行。它提供了一个巧妙的方法在 CSS 中使用 HTML 标签上的属性，在很多情况下都能帮你省去了以往需要 Javascript 处理的过程。

C语言实现合并排序

递归算法是把一个问题分解成和自身相似的子问题，然后再调用自身把相应的子问题解决掉。这些算法用到了分治思想。

其基本模式如下：

分解：把一个问题分解成与原问题相似的子问题

解决：递归的解各个子问题

合并：合并子问题的结果得到了原问题的解。

现在就用递归算法，采用上面的分治思想来解合并排序。

合并排序(非降序)

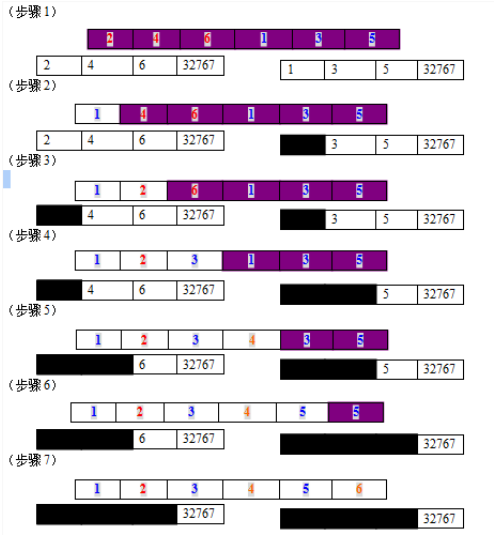
分解：把合并排序分解成与两个子问题

伪代码：

```
MERGE_SORT(A, begin, end)
if begin < end
  then mid<– int((begin + end)/2)
       MERGE_SORT(A, begin, mid)
       MERGE_SORT(A, mid+1, end)
       MERGE(A, begin, mid, end)
```

解决：递归的解各个子问题，每个子问题又继续递归调用自己，直到 "begin<end" 这一条件不满足时，即 "begin==end" 时，此时只有一个元素，显然是有序的，这样再进行下一步合并。

合并：合并的子问题的结果有个隐含问题，即各个子问题已经是排好序的了(从两个氮元素序列开始合并)。做法是比较两个子序列的第一个元素小的写入最终结果，再往下比较，如下图所示：

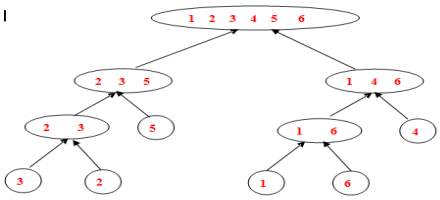


图中：待排序数组为 2 4 6 1 3 5

把 2 4 6 和 1 3 5 分别存到一个数组中，比较两个数组的第一个元素大小小者存于大数组中，直到两小数组中元素都为 32767.

这里 32767 味无穷大，因为 c 语言中 int 类型是 32 位，表示范围是 −32768−−−−−32768。用无穷大作为靶子可以减少对两个小数组是否为空的判断，有了靶子，直接判断大数组元素个数次就排完了。

整个过程中执行过程示如下图：



分解 + 执行时自上向下，合并时自下向上■

SQL Server子查询和表链接

1. 子查询概念

(1) 就是在查询的 where 子句中的判断依据是另一个查询的结果，如此就构成了一个外部的查询和一个内部的查询，这个内部的查询就是自查询。

(2) 自查询的分类

1) 独立子查询

-> 独立单值 (标量) 子查询 (=)

```
Select
    testID,stuID,testBase,testBeyond,testPro
from Score
    where stuID=(
        select stuID from Student where stuName='Kencery'
    )
```

-> 独立多值子查询 (in)

```
Select
    testID,stuID,testBase,testBeyond,testPro
from Score
    where stuID in(
        select stuID from Student where stuName='Kencery'
    )
```

2) 相关子查询

(3) 写子查询的注意事项

1) 子查询用一个圆括号阔气，有必要的时候需要为表取别名，使用 “as 名字” 即可。

2. 表连接 \

(1) 表链接就是将多个表合成为一个表，但是不是向 union 一样做结果集的合并操作。

但是表链接可以将不同的表合并，并且共享字段。

(2) 表连接之交叉连接 (cross join)

1) 创建两张表

```
use Test
go
create table testNum1
(
    Num1 int
);
create table testNum2
(
    Num2 int
);
insert into testNum1 values(1),(2),(3)
insert into testNum2 values(4),(5)
```

2) 执行交叉连接的 SQL 语句

```
select * from testNum1 cross join testNum2
```

3) 注解

交叉连接就是将第一张表中的所有数据与第二张表中的所有数据挨个匹配一次，构成一个新表。

4) 自交叉的实现

执行插入 SQL 语句 :

SQL Server 子查询和表链接 II

insert into testNum1 values(4),(5),(6),(7),(8),(9),(0)

执行自交叉的 SQL 语句：

```
select t1.num1,t2.num2 from testNum1 as t1 cross join testNum2 as t2
```

5) 另外一种写法：

select * from testNum1,testNum2 不提倡使用，首先是有比较新的语法，缺陷是逗号不明确，并且这个语法与内连接和外连接都可以使用，如果使用 join 声明，那么语法错误的时候可以报错，但是使用这个语法，可能因为部分语法的错误，会被 SQL Server 解释为交叉连接而跳过这个语法的检查

(3) 表连接之内连接

1) 内链接是在交叉连接的基础之上添加一个约束条件

2) 语法 :select * from 表 1 inner join 表 2 on 表 1. 字段 = 表 2. 字段

```
Select s1.stuID,
    s1.stuName,
    s1.stuSex,
    s2.testBase,
    s2.testBeyond
      from Student as s1
    inner join Score as s2
      on s1.stuID=s2.stuID
    where s1.stuIsDel=0;
```

(4) 表连接之外连接

1) 执行下面的 SQL 语句

```
create table tblMain
(
  ID int,
  name nvarchar(20),
  fid int
);
create table tblOther
(
  ID int,
  name nvarchar(20)
)
insert into tblMain values(1,' 张 三 ',1),(2,' 李四 ',2)
insert into tblOther values(1,'C++'),(2,'.net'),(3,'java')
select * from
  tblMain as t1
    inner join
  tblOther as t2
    on
  t1.fid=t2.id
```

2) 在内连接的基础之上，在做一件事儿，就是将 tblOther 中的 Java 也显示出来，这时候就要使用到外连接，外连接有左外连接和右外连接。

3) 左连接和右连接有什么区别呢?? 区别就是 ** 连接就是以 ** 表为主表，在内连接的基础之上，将没有数据的那张表的信息还是要显示出来供用户查看，那么这个主表就是要显示的那张表。左外连接和右外连接的分别是在前面的这张表就是左表，在后面的那张表就是右表，左连接使用 left join , 有连接使用 right join。

4) 上面重新执行下面的 SQL 语句，就会显示出 tblOther 表中的 Java。■

编者按

在《程序员之拍案惊奇 :为什么我会一天到晚的想说 FUCK ！》这篇文章里我贴一张程序员抓狂的配图，其实这一点都不夸张，读读下面这个故事，我相信无论谁做这个代码审查的当事人都会抓狂，你觉得呢?

与一个印度外包Java技术负责人的对话

在《程序员之拍案惊奇 :为什么我会一天到晚的想说 FUCK ！》这篇文章里我贴一张程序员抓狂的配图，其实这一点都不夸张，读读下面这个故事，我相信无论谁做这个代码审查的当事人都会抓狂，你觉得呢?

这是一个真实的发生在 Java 代码审查中的故事。

被审查的是下面这行代码 :

```
if (currentQueryType.name().equalsIgnoreCase("ALL_THE_WORDS")) {
        ...
 }
```

其中 currentQueryType 是枚举，在其它地方定义，代码如下 :

```
public enum QueryType {
        BOOLEAN, DOCUMENT_IDS , ALL_THE_WORDS, ANY_OF_THE_WORDS, LITERAL_PHRASES;
 }
```

审查者 :

(心里想 :什么玩意?) 请使用 switch-case 语句重构这段代码

印度外包技术负责人 :

这样写不行 :

```
switch (type.ordinal()){
    case 0:
   ...
    case 1:
    ... }
```

审查人 :

???

像这样写 :

```
switch (type) {
    case DOCUMENT_IDS:
    ...
    case ALL_THE_WORDS
    ... }
```

外包技术负责人 :

这样也不行 :

审查者 :

肯定能行，让我看看你的编译输出信息

外包技术负责人 :

我想原因可能是我们在 switch case 里使用了 === 操作符，而在 if/then/else 里我们使用 == 进行比较 :点击

审查者 :

我们不是写 Javascript，是 Java !

外包技术负责人 :

但我这边的 switch case 是这种情况 :所有的 case 它都认为是 ture，都去执行，而不是只执行等于我传入值的那个 case，比这个值大的它也执行。如果我传入 2，case 2 会执行，case 3 也执行，我能把代码发给你吗，你可以在你机器上试一试。

审查者 :

你是不是忘了在每个 case 后写 break; ?

外包技术负责人 :

哦。我在 switch case 前后都放了一个 break(断点)，这样我可以按 F6 进行调试。

审查者 :

我不是跟你说断点 (breakpoinit)，我说的是 break 语句！在谷歌里搜一下 switch case

外包技术负责人 :

哦 !!!

正能量系列：失业的程序员(一)

本文主要讲了失业程序员的一个创业过程，其实，在创业路上最艰难不是 Idea 也不是项目的技术难度，而是各种创业路上的”不相关障碍“。

不小心，我失业了。

原因是前几天和我的部门经理拍了桌子，我的组员去内蒙古出差，项目没有中标。年后，长得很像猪刚烈的部门经理发飙了，要辞退我的组员。

我纳闷了，我的组员是技术支持，要退也应该退销售啊。不过我知道猪刚烈不敢。

我好说歹说了一个上午，甚至我都提到了“真、善、美”的思想，都没有让猪刚烈回心转意。一时冲动拍了桌子，提出我也和我的组员一起 " 西去 "。

其实是老板的亲戚 ----- 猪刚烈大笔一挥，准了。于是我和我的组员当天下午就递了辞职报告，收拾了东西走人。

该组员很感动，表示要继续跟着我干，而且是一定要好好干。我去哪，他就去哪。我也很感动，一拍脑门说”创业吧，再也不伺候任何人了“，组员呆呆的看着我至少有 64 秒，一砸拳，应了。

创业讲起来简单，但要真做起来还真不是那么一回事儿。

幸好我早期积累了一些人脉，失业第二天，我就开始公交一日游，和各个”朋友、领导、兄弟“都见了一面。不过平时一聚会就说手上项目多的鬼一样的朋友，到真赤裸裸的求助时，个个也很赤裸裸。于是第一个月我和我的组员，一无所获。

猪刚烈时不时的打电话问候我，问我公司开在哪了，要送花篮给我。我只好打着马虎眼说快好了、快好了，如妥当第一时间告知，猪刚烈这才得意的挂掉电话。

说实际的，我多年的积蓄加上组员的积蓄还不足以立马成立一个公司。吃饭是首要问题，我们和大多数初期创业者一样，先从野活干起。

我擅长的技能是 Java，同时有着 2 年的 Andriod 的开发经验和案例，在还没”西去“的时光里，我在公司是公认技术 No,2 (No1 是猪刚烈，没人敢称第一)

目标在哪？没有，我的首要目标是解决两张嘴的问题。

到了第二个月，我好不容易从大学学姐的单位接了一个页面 UI 改造的活，由于没有公司不能开票，只能以劳务费的方式支付，该活的金额不比”绝世奇书 --- 降龙十八掌“贵多少。

第一桶金如获至宝，最激动和兴奋的是我比我还小 5 岁的组员，一个晚上通宵就把该活搞定了，我刮目相看，看来干自己的活，效率就是不一样。收到款时，我把 80% 都给了组员，除去请学姐吃了个饭，我自个剩了一个月的手机话费钱。

我向组员宣布，这 已经向我们创业成功迈出了一大步。组员表示真心认可。他说现在有信心。

我苦笑，所谓初生牛犊不怕虎。

由于学姐的给力推广，我连续又接了 2 个页

正能量系列 :失业的程序员 (一) II

面 UI 改造的活。组员很有激情的搞定了,其动力和精神无比刚猛。我突然开始佩服起他们。

虽然前景依然很黑暗,但是至少我和组员”不饿死“的问题已经 OK 了。

我开始大量的和朋友沟通和取经,虽然有的”经“着实很坑爹,但是也收益匪浅。期间不乏有很多来自其他公司的”看起来还不错“的 Offer,我拒绝了,我认为不能对不起我的组员。

接下来的日子,陆续又来了几个 UI 改造的活,我全部交给了组员。而我,窝在家里开发了一个我思考了很久很久很久的项目,叫做网站内容监测小助手。大致的用 途是,用户可以指定选择一些网站,用来分析这些网站日更新频率、分析关键字、获取最新内容概要以及行业相关指数监测、同行网站指标分析等等。

我对我思考了很久的 Idea 很有信心,组员呆呆的看着我的思路,也说”蛮有信心“。

没几天后,我原来公司的一个女组员 (程序员) 给我打了电话,因为受不了猪刚烈的变态管理,也走人了。出来后打算跟我。我没拒绝。

有了三个人分工就比较明确了,我负责 Idea 的主要开发,女组员负责辅助我。同时如果男组员的 UI 改造活比较繁琐,女组员需要临时发配去辅助男组员。

也不知道为啥,自从做了学姐的 UI 改造活后,我发现这个很多高级程序员看起来不屑一顾的活也是很有意思的。学姐的单位是制造型企业,这些同类企业都需要门 户,但是我们不要认为这些企业门户还没有要等着我们去建设,大部分都已经 OK 了。但是这些企业的老板日进斗金都不愿意花钱去改造这些网站,做的最多的是 UI 界面的改造,只需花 20% 的钱又可以看起来好像重做了一个网站。也就是因为这个并且经过学姐带着我到多家企业的忽悠,这些企业果然轮番进行了 UI 改 造,有的企业收到我们的货后,表示下半年还要”装修一次“。

商机是无限的,虽然有大也有小。我几乎死皮赖脸的从我亲戚那借了一些钱,和两个组员共同成立了一个”工作室“。我占 49%,其余他们平分股权,工资很低很低很低。暂时的主营业务就是”擦屁股“---- 为各大已经拥有门户网站的传统型企业改造 UI。

男组员干的不亦乐乎,有时我路过他们,会听到男组员给女组员讲述未来的梦想,如 " 上市 " 等关键字。我听了差点疯掉,但是回过来想想也很佩服,创业的首先基础是有一个梦。没有梦,那就是什么都没有。

其实,创业路上最艰难不是 Idea 也不是项目的技术难度,而是各种创业路上的不相关障碍。

这天下午,女组员突然来告诉我,她”有喜“了。我尴尬的表示恭喜,但同时我也知道我们要给”老弱孕残病“让个座。

3 个月后,女组员离开了我的团队。我请大家吃了个饭,女组员噙着泪水表示等她完成”造人工程“后,还要回来。

于是,又剩下我和男组员的奋斗。我继续着我的 Idea, 男组员经历了这么多个实际”项目运作“后,也逐渐变得成熟起来。

猪刚烈又打来了电话,我没接。但我知道我总有一天会接的。■

编者按

本文主要讲了失业程序员的一个创业过程，其实，在创业路上最艰难不是 Idea 也不是项目的技术难度，而是各种创业路上的”不相关障碍“。

正能量系列：失业的程序员(二)

闹钟响！

迷迷糊糊的我砸了一下开关，竟然把闹钟砸坏了。

昨天接到学姐的电话，说是帮我介绍了一个钢管制造厂企业型宣传网站的业务。难度不大主要是美工活，金额还没定，学姐自认为不会少。并约好了晚上一起请该企业一位姓童的副总吃“便饭”。该副总是学姐公司的一个大客户。

挂完电话，组员的眼睛呈绿色状。我知道又碰上他的强项了。

他绿我可不能绿，我知道这是我进行创业后的第一次“应酬”并且还是主人。我满怀希望的打开 度娘 搜索临近经济实惠又不能太土的饭店，一边大脑飞快的计算请客吃饭的预算和该项目的金额到手后是否能成正比。

正当我找的满头大汗、后背发凉的时候学姐给我来了电话，说是饭局改成了钢管副总请客，同往的还有几个据说也是其他企业里头和脸同时具备的人物。

我庆幸逃过了一次无法预计的开销。

学姐让我们穿正规点。

我找了件我爸的西装，偏大，套上后组员说我像木偶。组员不知道从他外公还是姥爷那找了一间中山装，我差点撕了他。我只好又从我爸那借调了一件西装。

晚上，两个木偶在学姐极端怪异目光的带领下走进了一家很具档次的饭店包厢。

学姐开始轮番介绍。我们极度拘束，同时也知道了坐在最中间的那个瘦老头就是钢管。

我和组员也被“头和脸”们喊成了带总字的称呼。我很飘飘然，偷偷瞄了一眼组员，他已经进入了梦境般的陶醉表情。

到现在我才知道，这种场面的应酬很具有中国特色。

在学姐的带领下，几乎从来不喝酒的我和组员轮番敬酒钢管和“头和脸”们，钢管连问都没问就夸我们年轻有为，将来必成大器。搞得我和组员兴奋到极点，一时间忘记了手中拿的到底是水壶还是酒壶。

敬酒高潮过去了。组员赶急慌忙的说去上厕所了。

喝成如此状态我还是第一次。钢管在我眼里变成了重影。

“刘总啊，这次我们厂的网站有信心做好吗？”钢管在我没有任何准备下突然发问。

"放心吧，童总。我们这次一定全力以赴，保证让你们满意！ “ 这是我的原话。其实我心里此刻最真实的想法很俗，就是到底做一下多少金额。

”技术我是不懂，我只是希望做的要好看，

正能量系列 :失业的程序员（二）II

要能反映企业的形象，让别人看起来是国际性企业。“ 这是钢管的原话。

这里我打断点评一下 ：

1、貌似很多客户都以不懂技术为荣。动不动就以此作为口头禅，甚至猪刚烈以前带我见客户时也这么说。我作为新时代木偶实在不知此为何故。

2、” 好看 “是网站的潜规则，不过到底什么是好看。很多客户自己也不知道。

3、我和组员都认为 宣传性门户网站 是一个非常好做的网站。事实上后来我们才发现，做任何事情都是不简单的，带着浮躁的心态做不好任何事情。当然这是后话。

---------- 以上点评结束

此时，组员摇摇晃晃的回到餐桌。表情呆滞。

” 童总啊，你总不会连我都不信吧。他们公司可是做过好多类似企业门面的哦。我带他们来也是本着认真负责的态度的。“ 学姐开始发挥了作用

我很感激的看着学姐，我预计学姐来之前给钢管做过不少功课。此次，我再一次感叹人脉的重要性，和客户之间的信任度很大程度决定于你的人脉。

组员又赶急慌忙的跑去厕所。

“哪里，哪里。我只是关心一下嘛！你可要帮我把把关啊！” 钢管果然吃这一套。

又是一阵觥筹交错。 我没一会儿也开始跟着组员接二连三的往厕所跑。

饭毕。我坚持不要学姐送。我扶着烂泥状的组员步行回家。

组员断断续续的讲话。开始我怎么也听不懂，后来终于听明白了。大意是我喝多了要我早点回家休息，晚上他要加班！

我眼泪突然夺眶而出。

。。。。。。。。。。。。。。。

花絮 ：

1、钢管的网站完成的还算顺利。金额不多，但是因为学姐的人缘，比一般性同类网站的收费高出一小截。

事后经过我认真估算后，把多出来费用的用支付宝打给了学姐，学姐扣除了当天她的打车钱又退给了我。说 :" 创业初期艰难，希望我们好好努力。他日成功后再回报也不迟 "。

我当时感动的想撞墙。

2、钢管的网站是我和组员的工作室 盖了章的第一个合同。我打算 弄个镜框裱起来。

3、我的 Idea 开发了有 60% 了。我在等待和努力着。不放弃就是一种成功。

4、我和组员也是从那次之后开始真正的 无缝配合。我后悔当时在猪刚烈蹄下上班时没有好好重用他。瞎了我的狗眼。■

专题：响应式Web设计应到来

链接：http://developer.51cto.com/art/201303/384588.htm

正能量系列：失业的程序员(三)

作者 / 沈逸

（一）

这段时间我去参加了一个管理培训班，说实话去之前真的很痛苦，我一向认为那些都是骗人的玩意儿。

在 qq 上找我学姐吐槽，说现在的广告真烦，搞这么多培训班谁去啊。学姐没回我，过了一会一个电话打来了，很强硬的建议我去一次，说去一次你会有所收获的，并且别忘了给她发几张会场的照片，挂了电话。

我不敢拒绝学姐的建议的原因，地球人都知道。给学姐发照片的原因我太清楚了，防止我没去说去了。

我从垃圾邮件中找了一个免费的就近的培训班，该培训班主题是产品开发团队的管理。属于试讲，在某酒店举行，而且第一次还免费，我报了名。组员自从上次饭局后买了一件劣质西装，我借了，组员用生死别离的表情告诉我千万别把饭菜油滴在上面，千万别把烟头在上面烫个洞，千万别。。。。。，我“摔”门而去。

到了会场，已经来了 N 多人，从外表上看都是蛮年轻和我差不多货色的人。我很不屑一顾，我只知道赶紧用手机拍两张照片，然后拂袖而去。

咔嚓、咔嚓，完事收工，我正打算走人此时主持人宣布“上课”了，会场立马安静并且熄灯了，瞬盲的缘故我摸了好一会儿才找到会场门，推了几下纹丝不动，我又很刚猛的推了几下，涛声依旧。我 了，我估计这是主办者故意的，不听完你休想走。

既来之则安之，我索性找了后排一个位置坐了下来，看会议安排也就两个小时，听完回去公车还有。

听了一会儿，说真切的，我个人觉得“老师”讲的很扯淡，也不从易入难，上来就是 10 个人的团队，后来上升到 20 个人乃至 50 人以上的团队管理法则、如何有效沟通、如何精细分工、如何约束研发人员、如何利润最大化，老师 讲了才半小时，就已经上升到上市企业的团队管理了。为何不讲讲 2-3 个人团队是怎么管理的呢，我真的不信到会的都是大企业家和上市公司 CTO。

我旁边有个瘦高个，不住的在频频点头，表情极为认可。此时还有人举手问问题，大意是现在他负责的团队有 20 个人左右，如何在业务和市场扩张很大的情况下，节约成本为前提更加有效的利用团队资源。

我擦，我忍不住说了句要被和谐的话，都做到这规模了，还来听啥课啊。

旁边的瘦高个看到了我不屑的表情，凑了过来问我是否现在手下团队成员比这还多。

我为了不被瘦高个看成”矮胖挫“，连忙点头称是，说我现在手下也有 30 个人了，业务开拓的太快，在团队管理上很头疼。

瘦高个立马把手伸了过来和我握了握，并递上一张名片，我一看，我市某知名软件公司的项目总监。据我所知，该企业 300 人以上。

看来是遇到真人真事了，我赶紧递上我在家门口复印店花 38.6 元印的名片，职位是 XXX 副

正能量系列 :失业的程序员（三）II

总经理。这个名片印法是学姐教我的，千万不要印一把手的头衔，我包里甚至还有业务经理、项目经理、技术总监等头衔的名片，因为在俺们国家，谈项目时”副总“说的话可以算数也可以不算数，这个你懂的或以后会懂的。碰到牛人时一般要递上职位对称的名片，否则人家不是借口尿检，就是跑出去离你很远接电话，然后再也不回来。

接下来，台上老师讲啥我已经听不清了，反正老师已经讲到跨国公司的管理，我臆测这些事不会降临到我身上。

我至少以前跟着猪刚烈也混过一段时间，见闻见事还是很多的。我与瘦高个非常有素质的小声交流心得，瘦高个对猪刚烈扮演的我的一些事迹频频点头认可，我很享受这个过程。

没想到就这么交谈着，时间过得很快，一会儿两个小时结束了。老师宣布下课，接下来是自由交流时间，期间我和瘦高个已经握了无数次手，我甚至想递烟给瘦高个，被边上的保安严厉制止了。

此时会场热闹起来，大家都走下了位置，开始互相交流起来。瘦高个把我带到一些貌似瘦高个朋友的人群众，给我冠以”我市后起之秀“，”未来我市软件的希望“，”即将成为软件业中流砥柱＂等一系列听了激动的想撞天花板的名号。很快，我的手就被握爆了，我包里的名片破天慌的第一次发这么快，一会一盒“双层特惠名片”就发光了，我重复着猪刚烈扮演的我的个人事迹，大家都很“素质”，假如说到一些不被大家认可的事迹，首先他们会肯定这个做法的好处，然后极度委婉的指出还有更好的办法可以实现。

我认为，奥巴马和各国领导人交谈亲切会谈，也就热烈到这程度。

会议结束后，我发现会场大门是用拉的。

瘦高个表示希望经常沟通，说不定会有个项目需要以外包的形式合作，我很认真的把瘦高个的手机背熟了。

.....

（二）花絮篇

这次培训会下来，我确实如学姐所说有很多心得，我列一下，求大家共勉。

1、老师讲什么其实并不重要，很多都是照搬照抄的，这点老师自己也非常清楚。

2、这类培训最重要的是能接触到各个不同层次的人，开阔视野和积累人脉，很多今后的机会都是这么来的。所以说创业大家不要只集中在产品和技术上，积累人脉才是最重要的，我想瘦高个和其他精英都是这个目的，傻子才会花 2 个小时去听在网上随便搜搜都能搜到的教程，有教程的还带视频，看完视频还能看 2 分钟双层汉堡特惠套餐的广告，多带劲，多养胃。

3、创业除了技术和产品，自身修养很重要。（以下是作者梦话般的吐槽：求某些大神不要再把修养和技术水平划等号了，那些对于初期小型创业者来说很容易被误导，很多初期创业者成也成在技术上，死也是死在了技术上，很可惜。积累人脉和建立与客户的信任感，口才和表情很重要。如果语无轮次或者表情略带猥琐感不能跟随大家的话题的变化而变化，那肯定很快会被大家踢出圈子。■原文未完，请参考

http://developer.51cto.com/art/201303/384269.htm

冯大辉的这十五年：一个非典型程序员的回想和思考

冯大辉，一个嬉笑怒骂、犀利、尖锐乃至幽默的男人，但是也许不会有多少人知道他的过往和心路历程。

他是互联网圈子知名的技术专家，现在活跃于微博专职吐槽，并且在微信上希望用小道消息拯救中国的互联网。他就是冯大辉（@Fenng），一个嬉笑怒骂、犀利、尖锐乃至幽默的男人，但是也许不会有多少人知道他的过往和心路历程：

·大学进门就读生物学专业，却半路出家钻研编程技术，并自学成才；

·最初进入阿里巴巴时，他经历了两个月的高强度工作差点以为自己会猝死；

·他对现在程序员的心态的隐忧和期待；

·他对阿里云服务的恨铁不成钢以及对百度呼唤“狼性”的理解；

·他对中国互联网的关心和认识。

图灵社区对冯大辉进行的专访会让我们对这个非典型程序员有更全面深入的认识，TECH2IPO 对原文有删节。

从生物学到阿里巴巴

我要是有那么多钱，不是想吃什么就吃什么了吗

我没有经过高考，保送上的大学。当时其实有点被忽悠了，大家都说 21 世纪是生物学的世纪。但是上了学之后，我就知道，这根本不可能是我们这代人的未来，我当时也没有体会到什么乐趣。

一类人就是（有）苹果机或学习机的一代，我就属于（什么机器都）没有一代。我接触计算机其实非常晚，是从大学才开始的。

Linux 在当时只在小圈子流行，那时候业界流行的技术是 Delphi，PB，“PB 程序员，月薪万元”，那时候月薪万元是什么概念啊，我每个月的伙食费才 2、3 百，我要是有那么多钱，不是想吃什么就吃什么了吗（笑）？

我赚的第一笔钱是帮别人做的一个浏览器上用来管理的程序，虽然只有 1000 块，但那是我认可自己挣的第一笔钱。**我的朋友都震惊了，纷纷表示：靠，搞 IT 真 TM 挣钱！从那之后，我决心以后也要靠这种方式工作，也要通过这个方式赚钱。**

我觉得如果一个人要被称为码农的话就必须要在开发团队中做事情，要把我归纳进来的话，完全是滥竽充数。但是我喜欢这个圈子，也喜欢在这里做事儿。我关注这个行业，关注码农，关注这里面有趣的人。虽然完全是野路子撞进这个圈子，但是在每个环境中，新天地都不断地打开。

冯大辉的这十五年 :一个非典型程序员的回想和思考 II

在支付宝的第一年差点没累死我

后来我去杭州观察了一下,阿里巴巴不象当时网上流传的那样 (当时有很多负面的信息),是个很有生气的企业,很欣欣向荣的感觉,从环境到人,给我的印象都很好。于是这次下定决心,走吧。

当时支付宝属于急速扩张期,很着急在招人,而整个阿里也没有几个 DBA,这些人每个人手里还都有一摊事。我刚到杭州,还一头雾水的时候,就要马上开始高负荷的工作。后来我女朋友一个月之后到了杭州,她不知道我现在忙成这个样子,有一天我早上回到家里,一敲门,她开门看见我熬一晚上很疲惫的样子,哇一下哭了,说 :

咱们不干了!

我从三月到五月份两个月的工作,几乎一直都是这么度过的。当时的支付宝系统要从淘宝独立出来,自己独立建立一个全新的平台,阿里也从各个子公司调了很多专家进来,要在五月份上线。我从来没想过在阿里退休,但我想我有可能在阿里猝死。在支付宝的第一年差点没累死我(笑)。

回头想想,像我这样能力平常的人,怎么还会变成了别人眼中的行业专家呢 (请允许我假装一回吧)? 有些人的确有天赋,我可能靠的就是卖力工作、拼命把所有事情搞清楚、认真地去做。

现在很多人的世界观已经被完全颠覆了

有些人提倡八小时工作制,有些人提倡加班,其实这些在一定的场景下,或者说对某一类人来说都是合理的。新一代的程序员和我们经历的事情不一样,**虽然我很反感那些所谓的“感恩”论调,但是现在的年轻人很多都需要哄着来工作,可能这也是一种一代人看不惯下一代人的心理吧。**

我们当时崇拜的都是像求伯君、鲍岳桥,简晶这些人,或者搞开源软件的这些牛人,当时这些人里谁赚的钱最多? 没人知道,也没人关心这些不相干的事情。所以尽管我努力尝试去理解不同时代的 IT 人,但是很多时候仍然无法完全理解。

从我大学毕业到现在,大概有 15 年时间。这个社会的变化非常快,是一个大时代。我大学时候,那男女关系是多么纯洁,哪像现在这样礼坏乐崩(笑)。现在很多人的世界观已经被完全颠覆了。人们一方面很向往美好事物,但是另一方面又要在现实中挣扎,和社会去斗,为了买房而奔波赚钱。

做不好就是要被骂,这都是天经地义的

比如 12306 或者电信的一些东西,这些基础设施类的服务就是要做好,做不好就是要被骂,这都是天经地义的。

如果阿里没有早期的内在驱动,后来怎么会变好呢? 每一次系统故障,我们都是如临大敌。第一次系统故障就是由于我的疏忽,半夜把系统搞宕机了,这都是我应该做好的,是我的责任,做不好别人骂我是应该的。

李彦宏说的“狼性”,他讨厌的东西我都可以想象出来。大公司里有一些人就是这样,干多干少都行,干好干差都可以,整天从这个会议室窜到那个会议室,看看他对公司的贡献,几乎等于 0。**百度现在属于那种很疲的状态,你踢一脚也没什么反应,我就这样,我躺着钱就从天上掉下来了。**

嬉笑怒骂看世界

阿里云不要再瞎折腾了

我很反感总是要尝试改变别人的人,总是想把想法强加到别人头上。我从来就不会把自己的想法强加给别人,而有些人一定要说服我。

比如说,如果我说创新工场好,他们就会说我拿到好处了,如果再过段时间,创新工场给我投资了,他们会说,“你看,我早说了”。但是我也会反思,有时我说的话的确会有负面的东西,以前就有人说“丁香园怎么找了这么个傻逼来管技术,我要是 VC 就不给他们投资”,我就觉得很好笑,多亏这样的人他不是 VC。

至于调侃阿里云的话,其实我很希望中国能出一家云计算的公司,这样很多中小公司都会受益,我希望阿里云不要再瞎折腾了,赶紧提供一些可靠的服务吧。

应该尝试开阔一下视野

长期做一种事情的人容易形成一种观念,只有在我这个领域牛的人才是牛人,别的领域的牛人都是狗屎,都不行,看不上。写前端的和写后端的,搞微软的和搞开放技术的,写 C++ 的,觉得 Linux 领域的牛人都不行, IT 行业里的这种隔阂非常大,所以吵架在所难免。语言之争什么的,市场的选择也不是你争论出来的。

很多程序员过得没有希望是因为他们的视野太窄了,除了看技术,就是看科幻,我建议他们多看看人文历史类的书籍,这样的书可以引导他们理解别人的内心,看看小说什么的也可以很大程度上补充他们看问题的角度。

程序员整天面对的就那么几个人,经理就是监工的、客户就是傻逼,每个人的角色都已经设定好了,如果没有更多了解,圈子就会越来越窄。应该尝试开阔一下视野。

一些有意义的事

我们做的事是半公益性质的

我喜欢给自己找一些看似正向的理由,当时去杭州,我说服我自己和家人的时候都是说电子支付是能改变电子商务的事情,是对社会有价值的。但是后来做久了发现,电子商务也主要是帮人赚钱,无法解决更多的问题。支付宝做得再好,也无法改变人的生死。

但是如果信息和服务可以做得更好,就有这样的可能。我就希望如果我们能更好地给医生提供信息,是不是可以让他们的医术和待遇更好一点,这样整个医疗行业是不是会更好一点。

中国的人口基数这么大,将来医疗肯定是个大问题,所以我们能做的事也是越来越多。而我们这些对信息了解更多的人为什么不来做这件事呢?我一直觉得我们做的事是半公益性质的,但是不商业化是不可能的,我们的商业化是底层的,不会影响到使用。

中国互联网自己会拯救自己

关于互联网,每天都会有大量的信息,真真假假的信息都有,多数信息都是不那么可靠的,倒是很多在私底下流传的信息反而更加准确。只有小道消息才能拯救互联网,有一点恶搞的成分,其实还有下半句,“只有小道消息才能拯救中国”…

别那么严肃,这只是一句玩笑话,没有什么能拯救中国互联网。中国互联网自己会拯救自己。也没有谁能拯救中国,也要靠我们所有人。

我现在的确花了一点业余时间在运营微信公众帐号“小道消息”,最初的目的就是想摸索学习一下微信的运营方法,看是否能够对我们的业务有帮助。

倒是有些一发不可收拾了。有的时候会夹带一点私货,发表一些我对个别事情的看法,我也不知道会坚持多久,既来之,则安之吧。■

经验谈：如何快速应对项目需求中的变化

编者按

大家在项目中，印象最深的估计就是“需求变更”了，这个词无时无刻让 coder 们紧绷着。祈祷着没有变更，但是总是事与愿为，不管是进行中的，还是结束后。

大家在项目中，印象最深的估计就是“需求变更”了，这个词无时无刻让 coder 们紧绷着。祈祷着没有变更，但是总是事与愿为，不管是进行中的，还是结束后。总会有这种那种的变更、优化等着你去支持。

那么如何应对这种变更的需求呢？笔者有以下几点个人观点跟大家分享和探讨。

一、项目开始阶段

在项目开始阶段，不要急于去写你的代码，不要为一开始拿到需求就想到时间进度问题。当你拿到需求时**更重要的是先消化好需求**，从中间挖掘出今后可能会存在的发展方向。

消化需求主要为以下几个方面：

1、先整体了解项目的背景，项目生存的环境是什么？

2、仔细了解整体的交互过程，挖掘出你的代码框架要如何设计？

3、对上面 2 点消化后，开始选择你的主框架或主库；在没有合适框架情况开始规划你的库结构。

4、思考项目部署问题。如何规划你的文件分布以及目录结构，方面日后的维护。

5、评估时间时预留风险（可能会发生）时间，可以参考一下三点估算法来评估。

三点估算公式：

Te=(To+4Tm+Tp)/6

To：基于活动的最好情况，所得到的活动持续时间

Tm：基于活动最有可能活动持续时间

Tp：基于活动的最差情况，所得到的活动持续时间

Te：预期活动持续时间

二、项目进行阶段

这个阶段估计是最头痛的阶段，有时候基于各种因素。经常听到的是“XX，这里需要调整一下”、“XX，这个流程这里因为 XX 原需求调整一下”等等类似的情况。然而，没有圣人，这种情况不管前期考虑的多完善，在执行过程是不可避免的。我们唯一能做的是——用最小的代价支持变化的需求。

这里分享以下几点经验：

1、底层公用接口设计要功能单一，不局限调用方式，方面业务层二次封装。

2、在业务层规划好公共接口。

3、做好底层接口的二次开发，方便在业务层的灵活运用。

4、解耦代码，各模块独立，尽可能降低交叉引用。

5、做到 UI 与逻辑分离，减少对 UI 的依赖。

6、经常回顾你设计的代码。看看有啥不适之症，即时做好调整。

7、确定关键路径（花费最多时间

的路径，也就是项目的最后完成期限时间），优先保障关键路径的开发；原则就是先修主线，后修剪枝叶。

三、项目上线后的优化阶段

这是一个长期的作战过程，除非你的项目“Game Over”了。

而且一些变化会让你始料未及，那么如何去快速支持呢？这里大部分依赖于上两个环节是否设计的合理了。

建议如下：

1、同项目启动阶段一样，先拿到需求，仔细阅读需求，不要急于下手。

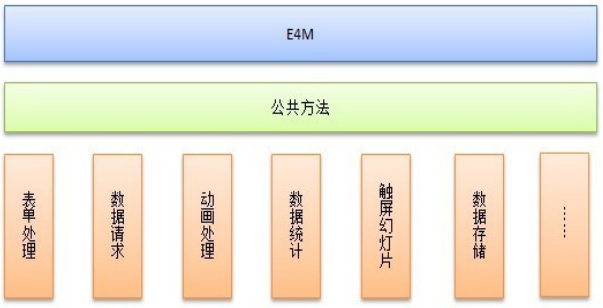
2、 了解交互的差异性，看看新的交互与之前的具体变化是什么，这里需要确认出来的信息是：a) 是否需要完全重构；b) 是否只是参数调整；c) 是否只需要屏蔽现在接口的调用。

3、 如果涉及交互大调整，需要重构的。这里就需要重新思考重构方案，不要急于直接重写你的代码。

四、两个示例

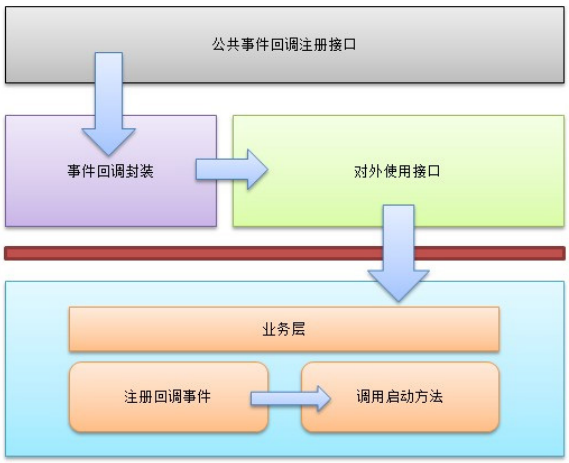
1、接口的设计

v 设计整体结构



整体结构设计

v 对单个接口调用和实现进行思考

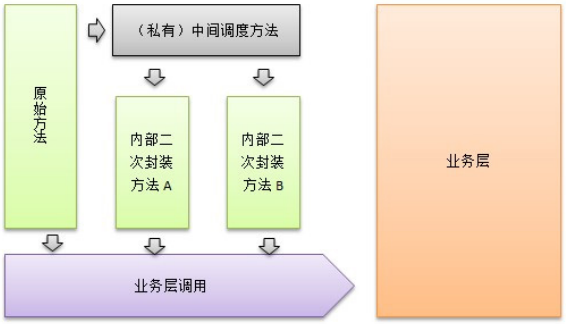


单接口设计思想

v 接口剖析：多样化的调用实现以及二次封装接口预留

s 事件注册句柄代码示例

s 接口方法实现代码示例



以 PageLoader 接口为例：

底层接口为 loadPage()

1、 HASH 加载请求处理：loadHash() -> loadPage()

2、 用绑定节点的方式处理：bind() -> call() -> loadPage()

■原文未完，请参考

http://developer.51cto.com/art/201303/382747.htm

几种华丽无比开发方式

今天我要说的，是几种看起来激动人心、华丽无比，但是可以让程序员们痛苦不堪的开发方式，特别适合那些热衷于折磨虐待程序员的项目经理和产品经理们。当然，掌握以后，偷偷用就好了，请不要来感谢我。

不要被我的标题骗了。我可不是来宣扬什么模型驱动开发，或者什么测试驱动开发的，那些都弱爆了。今天我要说的，是几种看起来激动人心、华丽无比，但是可以让程序员们痛苦不堪的开发方式，特别适合那些热衷于折磨虐待程序员的项目经理和产品经理们。当然，掌握以后，偷偷用就好了，请不要来感谢我。

进度驱动开发（SDD，Schedule Driven Development）

这是在国内最为流行的开发方式，大家心照不宣，口口相交，代代相传，我只是把它写下来而已。它最华丽的地方在于，可以百分之百，甚至百分之二百地压榨程序员的劳动力。

需要实现哪些需求？用什么技术？用什么平台？项目采用什么流程管理？这些都不重要。重要的是——什么时候交付？

假使说，老大们通知，下个月的这个时候要看到产品发布，那么：

三周以后就要拿出完备的产品准备上线；

两周以后就请发布 beta 测试版本，ST、IT 之类的东西就得在那之前完成；

本周就必须完成编码和 UT，那么周一设计，周二、周三开发，周四、周五测试和修正问题。

看，项目计划多么完美。项目时间本来就该是根据 deadline 倒排的。

项目做什么呢？先做那些相对重要的需求，可是如果时间紧的话就只好砍需求了吧……不！你怎么能那么容易就放弃呢？你看，我的完美的计划里面没有安排周六和周日嘛，大家可以来加加班嘛，年轻的时候不得奋斗一把嘛，不用砍需求，平时的时间再压一压不就可以如期上线了？

在热情洋溢的动员会之后，大家开始拼命赶工了，疯狂的一周过去了，测试团队始终等不到开发团队提供的发布包，难道“又”要延期了？

那还用问吗？当然！

测试团队的时间也是可以压缩的嘛。于是煎熬的两周过去了，发布日期眼看越来越不靠谱，项目经理觉得，他需要挺身而出了——

敏捷思想教导我们，搞不定的时候，质量不能丢、进度更不能丢，那我们只得砍需求了。这样，我们只发布“核心功能”总行吧……

可是什么才是“核心功能”呢？

对了，我们做完了哪些？要不，做完的就算“核心功能吧”？

太牛了！这真是一个伟大的创举！

别忘了，给程序员画饼也是项目经理重要的技能——大家再努努力，进度压力也是没办法的事，发布以后大家就轻松了，有好日子过了！

瞧，“没有发布不了的版本”，这是真的！

产品发布以后大家就轻松了，有好日子过了，这也是真的！

文档驱动开发（DDD，Document Driven Development）

这种开发方式也非常华丽，对于许多领导和老大们而言，文档胜过一切。架构文档要靠 ppt，因为他们的智商和知识不足以理解满是文字的东西，而胶片，则是最接近看图说话的好东西。设计文档，要靠足够详细的 word 文档，项目经理要看到你的文档细致到肯定可以轻松地指导编码，如果你明天突然拉肚子拉到抽筋，打嗝打到卡住，喝水喝到噎着，于是不幸住院的话，文档的威力就体现出来了，他可以轻松找到你的备份，替掉你的工作。

软件开发全套有十项文档，从工作任务书开始，只有完成了文档，你的工作才算完成。如果你要在邮件里面，或者会议上向大家传授一点什么技巧，你可得当心了，因为接下去劈头盖脸的就是这样一句“有文档记录吗？”，仿佛有了文档就有了一切，有了文档就买了保险——至于有没有人看，嗨，谁管呢？

别忘了，文档的核心地位需要贯彻到底。在绩效考核的时候，最能写的人，就可以成为优秀员工。代码这种无法体现智商差异的东西可以踢一边去，只有文档才是智慧和能力的综合代表啊。

指标驱动开发（IDD，Indicator Driven Development）

这种开发方式的华丽，源于它超强的数据化和量化的能力。写代码的目的是什么？完成需求？优雅设计？用户体验？你全错了。

再次强调，终极目的是测试覆盖率。

整个软件开发流程里，你可以找得到无数的指标要求，在做每一件事情之前，必须要像默念毛主席语录那样回顾一遍需要达成的指标，然后再动手。

有一天，你发现用户体验像屎一样的产品，居然自动化测试也可以达到 95% 以上的通过率，bug 居然可以收敛到 10 个 / 轮测试，而且 Findbugs /CheckStyle/PMD/Source Monitor/ Simian 之类的无数代码检查工具的结果页上，都齐刷刷地显示着绿条……

恭喜你，你成功了。

更重要的是，项目成功了。

装逼驱动开发（ZDD，Zhuangbility Driven Development）

这大概是几种开发方式中最华丽的一种。在设计前、写代码前，在做每一项事情之前，都要谨记装逼的重要性。对于很多不懂技术的领导来说，听起来越牛逼的软件，就越值得开发。

产品装逼：必须支持“云”和“大数据”，比如数据存储到服务端……■原文未完，请参考

http://developer.51cto.com/art/201303/383241.htm